

2026 State of Code Developer Survey report **executive summary**

The verification bottleneck

The software development landscape has shifted: teams aren't just using AI for experimentation—they're shipping mission-critical code. But our survey of over 1,100 global software engineers and leaders reveals a critical verification bottleneck. Code generation has accelerated, but review capacity has not, and for engineering leaders this represents a strategic risk. Leaders who break this bottleneck will convert AI potential into business value; those who don't risk drowning in unreliable code and technical debt.

The verification bottleneck

The central tension lies between the volume of code AI produces and the trust developers place in it.



The volume explosion

42% of the code developers commit today is AI-generated or assisted. They predict this will surge by over half in the next two years.



The trust gap

96% of developers do not fully trust AI-generated code. Also, 61% agree that AI often produces code that "looks correct but isn't reliable."



The review burden

38% of developers report that reviewing AI-generated code requires more effort than reviewing code written by their human colleagues.

This dynamic creates a bottleneck where code piles up or, worse, flies past code review. While trust is low, only 48% of developers are completely diligent about checking AI-assisted code before committing it, creating a significant margin for error in production.

Shadow AI and the governance gap

The verification challenge is compounded by a growing "bring your own AI" (BYOAI) culture.



The personal account problem

A significant portion of AI usage happens outside corporate oversight. Across the top 10 AI tools, an average of 35% of developers access them through personal accounts.



Fragmentation of toolchains

Individual-led adoption has led to a fragmented environment. On average, development teams now juggle four different AI tools with favorites like GitHub Copilot (75%) and ChatGPT (74%).

The toil paradox: Shifting, not shrinking

AI hasn't eliminated toil; it has shifted it to code verification.



Perception gap

While 75% of developers believe AI reduces toil, the data shows time spent on repetitive tasks remains static at roughly 24% of the work week.



The new toil

Frequent AI users report higher toil associated with "managing technical debt" and "correcting or rewriting AI code".

The technical debt dilemma

AI-generated code is a double-edged sword for codebase health.



New risks

88% report a negative impact on tech debt. 53% cite AI creating "correct-looking but unreliable" code. 40% cite code bloat through unnecessary or duplicative code.



New benefits

93% see positive impacts on tech debt. 57% report improved documentation, 53% see better test coverage, 47% refactor existing code.

A divergence in strategy: SMBs vs. enterprises

Organizations are responding to these risks differently based on size.



SMBs prioritize velocity

Smaller organizations report higher gains in time-to-market (76% vs. 67% for enterprises). But they pay a tax: SMB developers are also significantly more likely to cite "correcting or rewriting AI code" as a top toil source.



Enterprises prioritize control

Large organizations are 50% more likely than SMBs to have well-defined, distinct guidelines or automated checks for AI code (18% vs. 12%) and are 39% more likely to be rigorous about compliance reviews.

The solution: Vibe, then verify with SonarQube

If you have a high-trust, well-architected system with a robust testing and verification platform, AI will amplify your efficiency. But if you have a low-trust, chaotic system, AI will only amplify that chaos, giving you more low-quality code faster than ever before.

This is why Sonar customers find more success with AI. By implementing automated verification with SonarQube, teams break the bottleneck—allowing them to "vibe" with the speed of AI while automatically "verifying" every line.

Better outcomes:

SonarQube users are 24% more likely to report reduced vulnerabilities, 20% more likely to see lower defect rates, 18% more likely to see reduced rework costs, and 12% more likely to report improved overall code quality.

Higher effectiveness:

They rate AI tools as 19% more effective for assisting in new code development.

Reduced risk:

Non-users are 80% more likely to report that AI adoption led to more outages and incidents.