Sonar

State of Code Developer Survey Report

2027

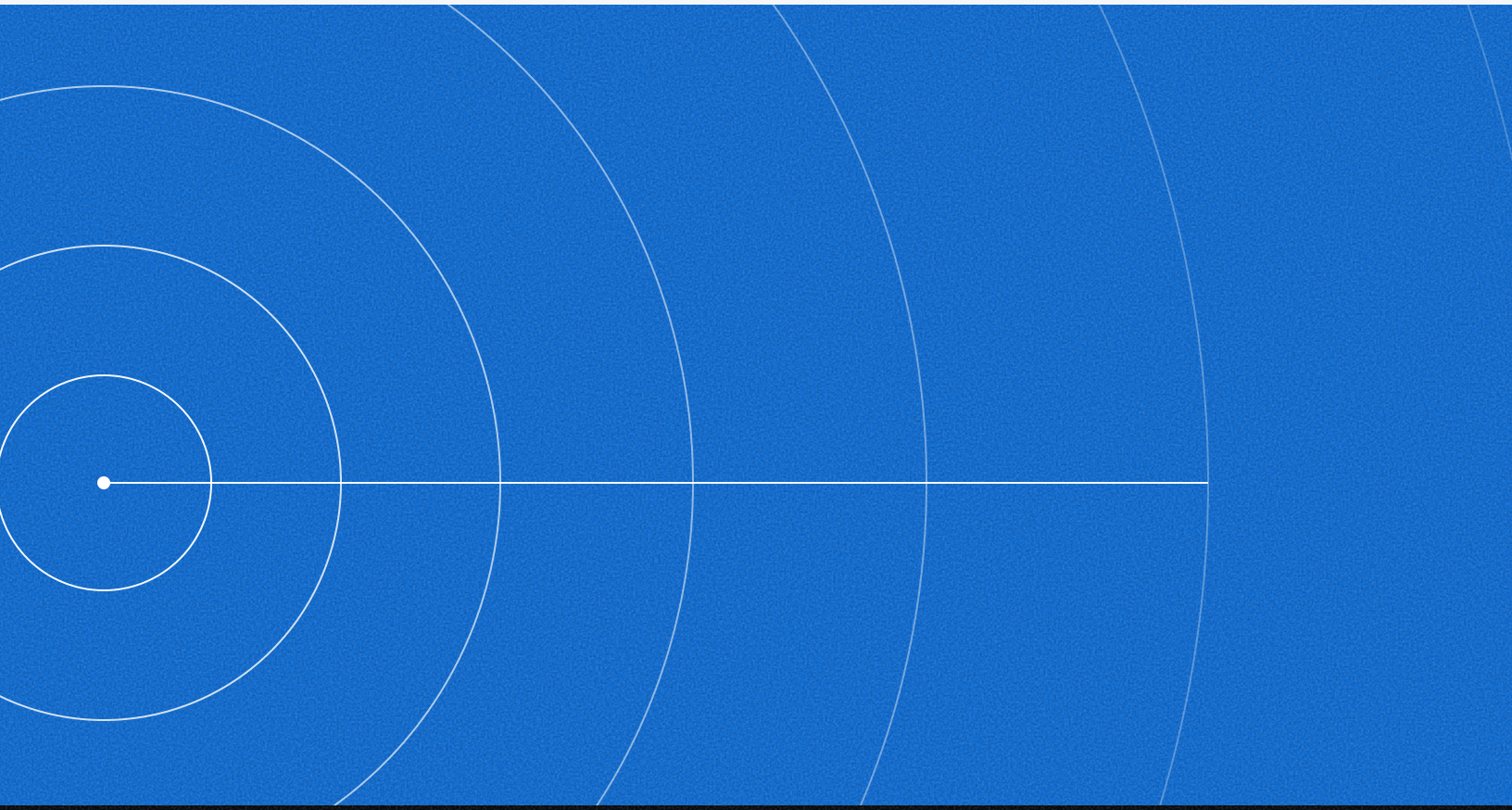


Table of contents

Introduction 03

About this report 04

Code is abundant, confidence is scarce 05

The software factory is coming fast 17

Security concerns dominate, even as more teams say AI reduced outage frequency 22

Governance is spreading, but few teams define it well 28

AI tends to make healthy codebases healthier and messy ones messier 35

Productivity is rising, but ROI is lagging behind 42

AI investments are shifting and diversifying 53

SonarQube: Ship agentic code with confidence 63

What leaders should do next 68

Appendix 72

Glossary of terms 76

Introduction

**Last year, code creation outpaced verification.
This year, agents have widened the gap.**

Our first State of Code Developer Survey found that the surge in AI-generated code had not yet delivered the productivity gains teams expected. It created a verification bottleneck instead, and the largest group of organizations still reviewed AI code the same way as code written by people.

This year, we asked over 1,000 developers and engineering managers what happened next. In short, agents made AI coding better, and code became abundant faster than teams could build confidence in it. Respondents say AI now generates or significantly assists 65% of what they commit, up from 42% last year and beating their predicted timelines by a year. Respondents rate AI as more effective now at most of the tasks we can compare with last year. This means they can use it in more of the software their businesses depend on.

But a chasm is emerging. On one side is the current state: code volume is way up, review fills the week, and 76% of developers and engineering managers say they generate AI code faster than they can verify it. On the other is where a third of respondents say they are heading within a year: software factories, where agents work autonomously on tasks. Our data suggests that getting from one to the other depends less on better models than on two things every development team can control: how they guide agents before they write, and how they verify every change before it ships.

About this report

The 2027 State of Code Developer Survey was a quantitative online survey of 1,026 professional software developers and engineering managers who use AI coding tools at work. Fieldwork was conducted in September 2026.

All respondents were 18 or older, employed full time or self-employed in a technology role, wrote code or managed software developers (or both), and had used AI coding tools for professional work in the past year. 83% write code (57% write code and manage developers; 26% say writing code is their primary responsibility), and 17% manage developers without writing code themselves.

Comparing with last year

Year-over-year comparisons use the 2026 State of Code Developer Survey (n=1,149, fielded 11 months earlier in October 2025 and released in January 2026). Both samples include engineering managers in similar proportions, with 15% who only managed developers in 2026 vs. 17% this year. We compare total with total and only call a change meaningful if it is statistically significant at 95% (two-proportion z-test). An asterisk (*) on the graph marks a significant change from last year's data, or a significant difference between the groups being compared. Where an answer option changed wording between years, we say so.

Percentages may not sum to 100% because of rounding or multi-select questions. Agreement figures combine "agree somewhat" and "agree completely" unless noted.

Additional demographic details, the analyst's note, and glossary of terms are in the appendix.

CHAPTER 1

Code is abundant, confidence is scarce

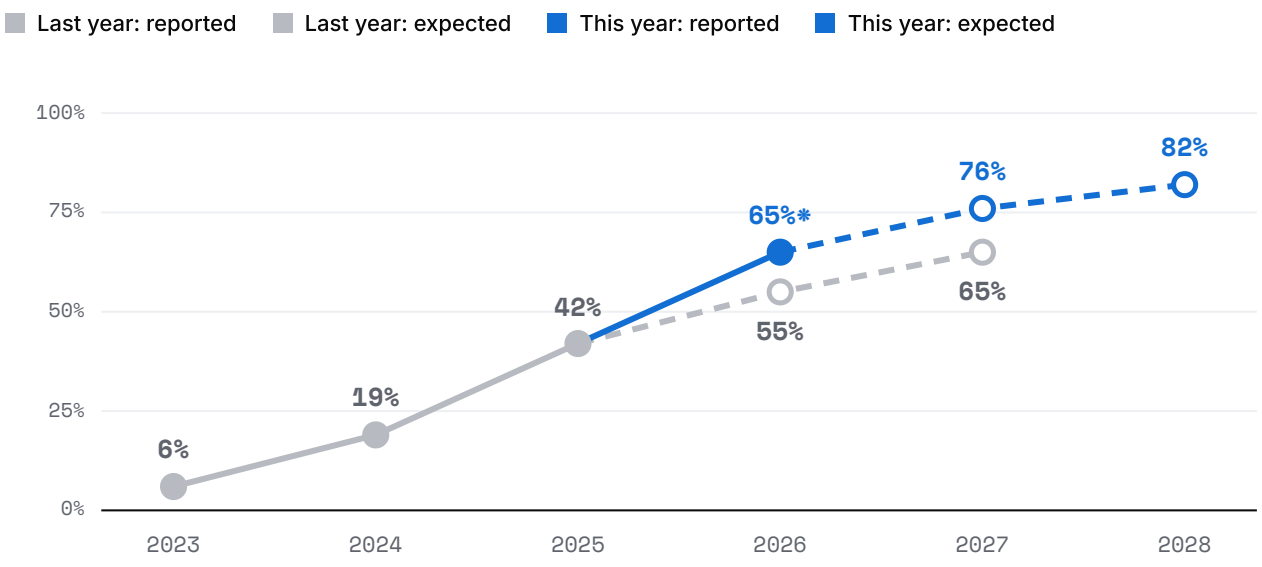
AI hit 65% of committed code a year early

A year ago, respondents told us AI generated or significantly assisted 42% of the code they committed. They expected that to reach 55% within a year and 65% within two. But it turns out they got to 65% in just one year.

A 23-point rise in one year is well above the 13-point rise respondents expected (42% to 55%). Generating code with AI is now firmly a mainstream practice.

AI code share hit two-year forecast in one year

What percentage of the code you personally committed or contributed was (or will be) generated or significantly assisted by AI tools / agents? (average)



Among respondents who write code · Last year n=979 · This year n=856 ·
*Significant change vs. last year

FIG. 01

AI coding is now used daily to write the software that matters most

85% of respondents use AI coding tools daily or multiple times a day, up from 72% last year. “Multiple times a day” alone rose from 42% last year to 60% now.

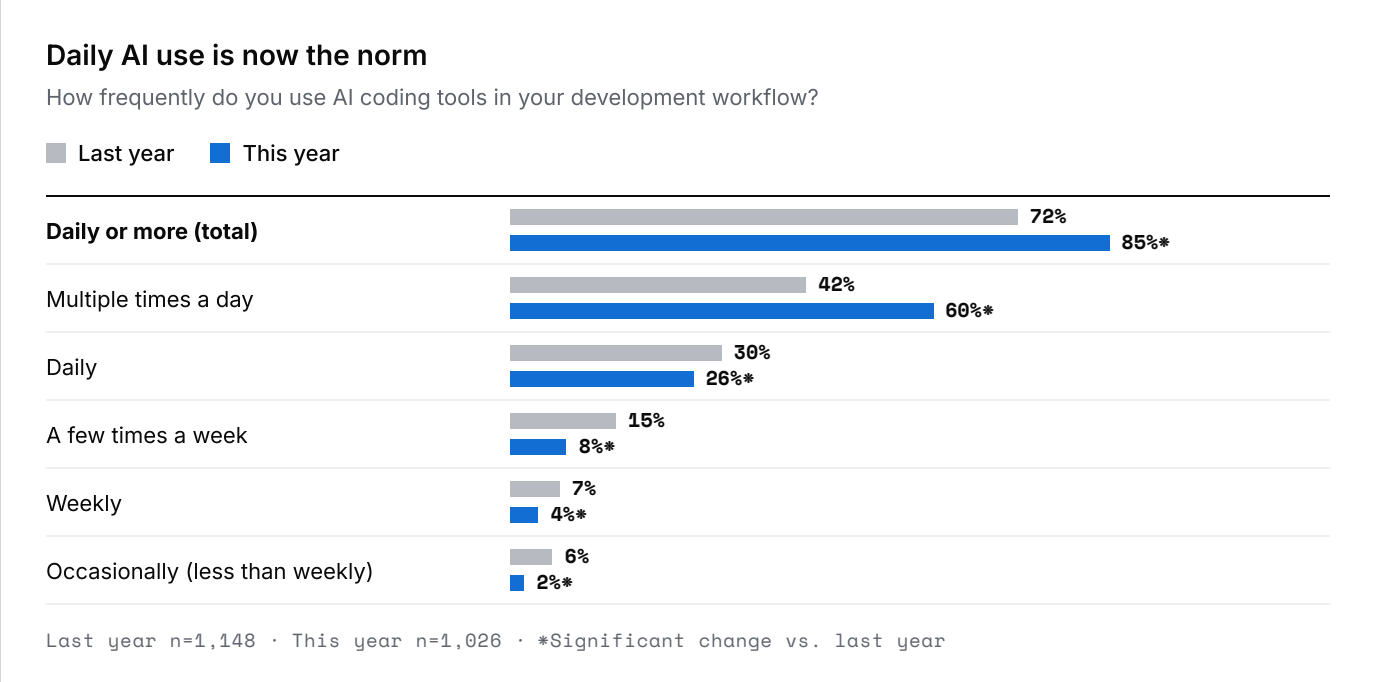


FIG. 02

More often, AI coding is writing software that businesses depend on. 76% of respondents say their team or company uses AI coding tools for business-critical or mission-critical services, up from 58% last year. Use for prototypes was already near saturation, so most of the growth went into production workloads, with customer-facing applications rising from 73% to 86% and business-critical services from 58% to 76%. This is a clear signal that overall comfort in the quality of AI-generated code is growing.

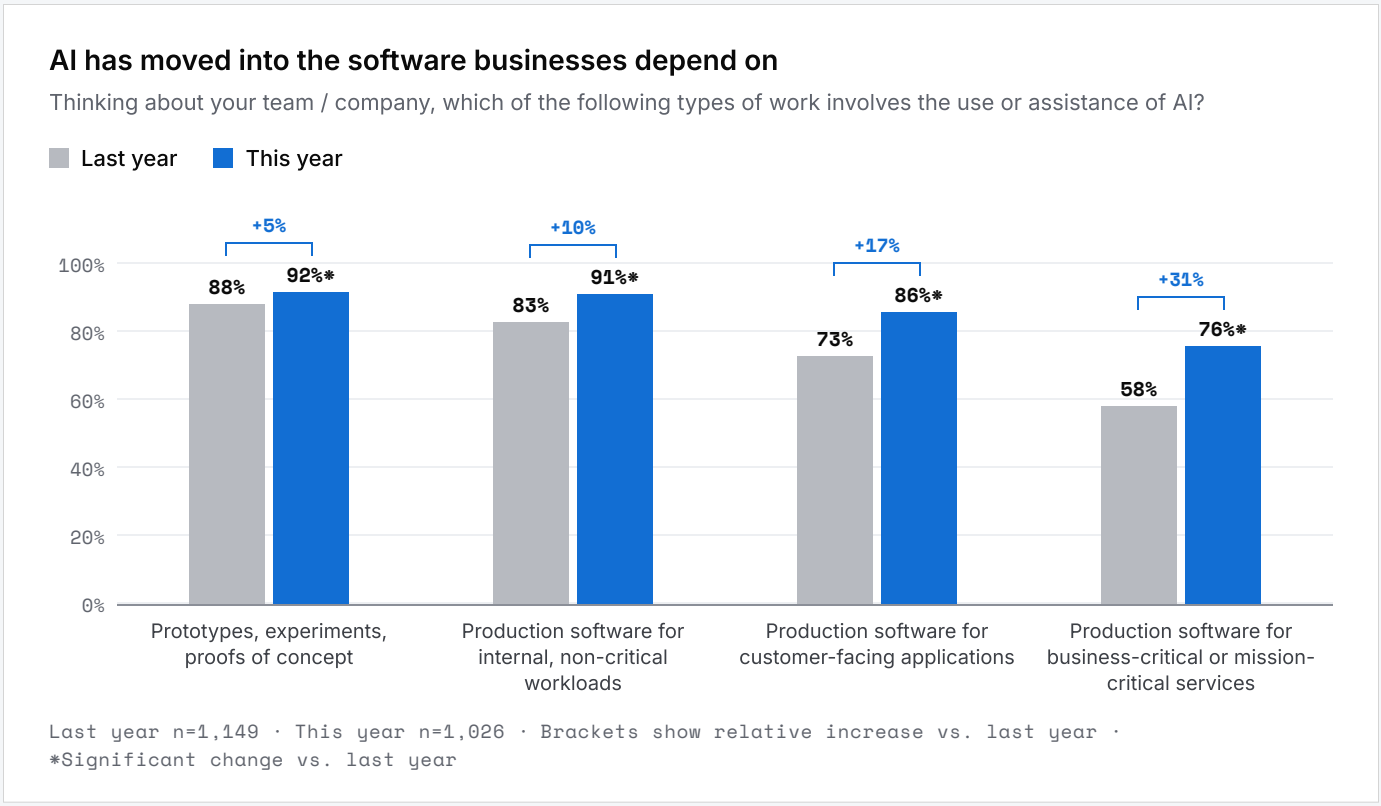
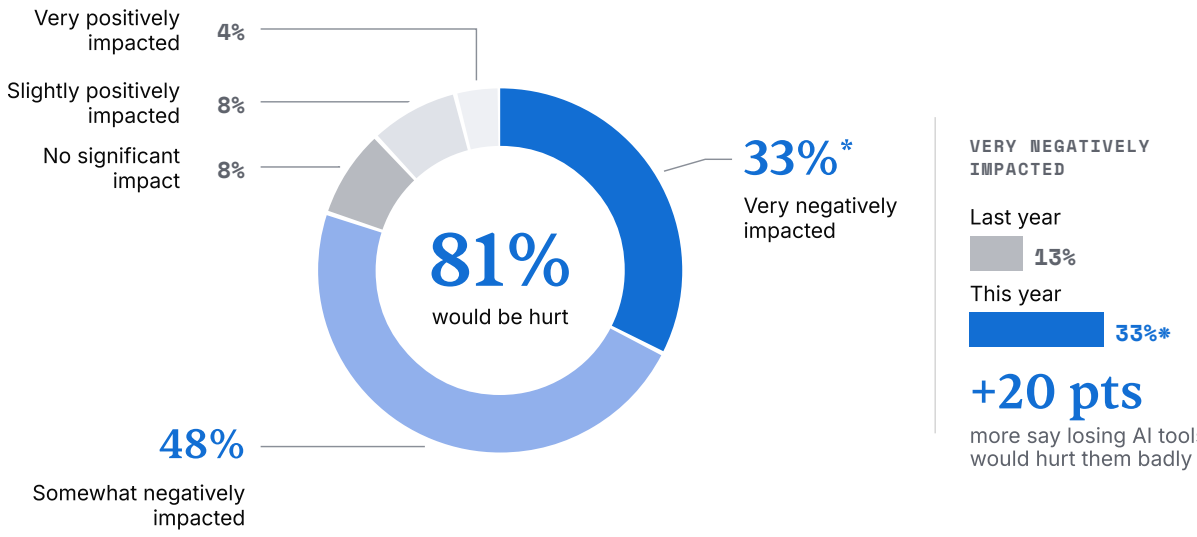


FIG. 03

Dependence has also grown. 81% of respondents say losing their current AI coding tools would hurt their ability to do their job, and 33% say they would be very negatively affected, up from 13% last year.

Dependence on AI tools has grown with use

Imagine you could no longer use your current AI coding tools / agents for your development work. How would this impact your ability to do your job effectively?



Last year n=1,149 · This year n=1,026 · *Significant change vs. last year

FIG. 04

AI has improved where it used to struggle

Adoption rose significantly for all tasks we can compare with last year, and effectiveness rose significantly for most of them.

By a fair margin, the AI use case showing the biggest gains in adoption was developing new code (primarily or solely written by AI). 48% of respondents last year reported using AI for this, but this year that number has jumped to 84%. Other use cases showing significant adoption gains:

- Translating code from one programming language to another: 50% to 71%
- Finding bugs (for example, stack trace analysis): 65% to 84%
- Code review: 55% to 73%
- Writing documentation: 74% to 86%
- Adding or updating functionality in existing code: 76% to 87%

AI writing the code is now the norm

For which of the following tasks is your team / company using AI coding tools / agents? · % using AI for each task



This year n=1,026 · Last year n=1,149 · *Significant change vs. last year

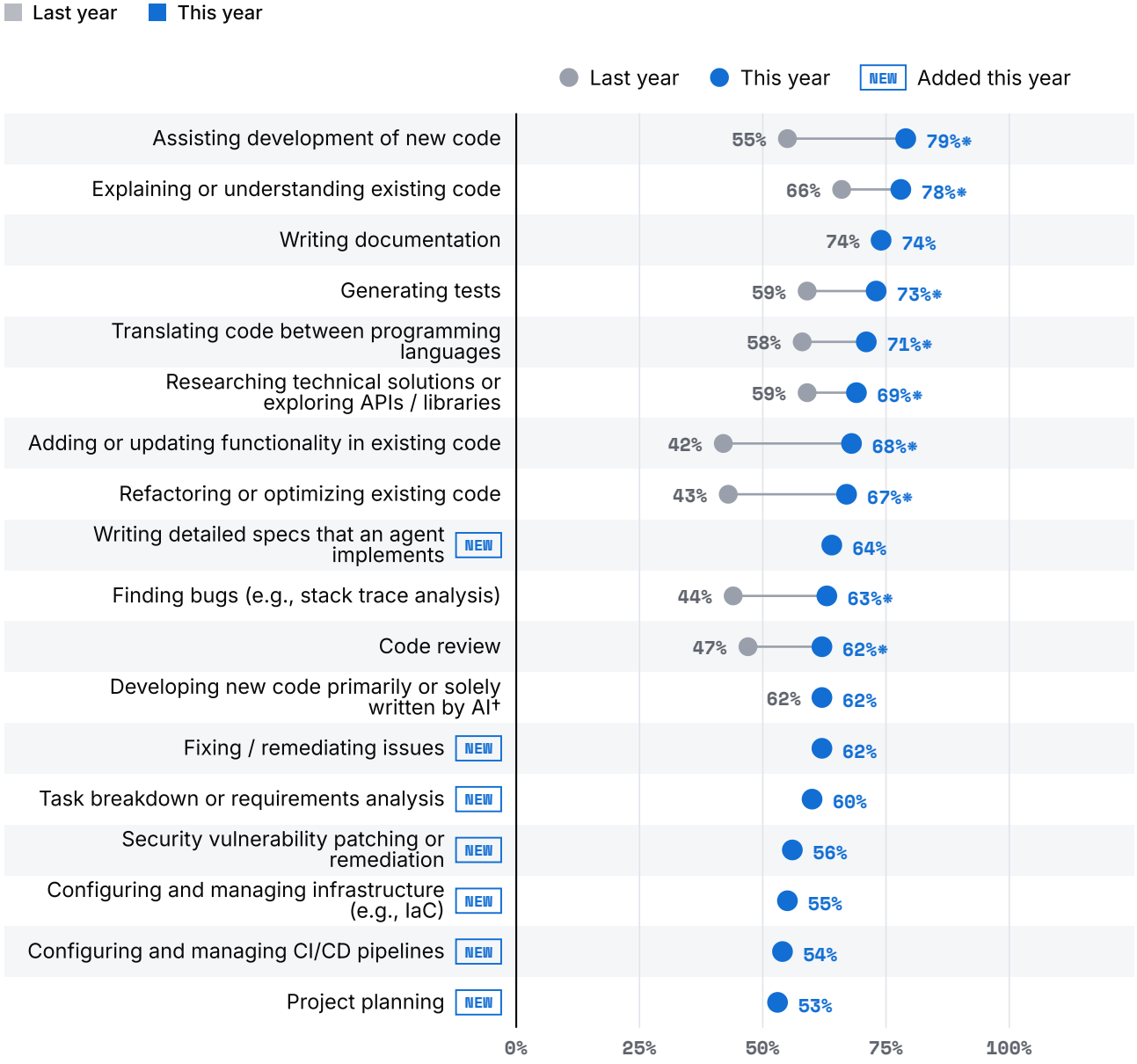
FIG. 05

Notably, the use case showing the biggest adoption gain (developing new code with AI) stayed flat year-over-year (YoY) in effectiveness. Most others improved YoY, and the largest effectiveness gains came in work on existing code (where AI was weakest last year). Among respondents whose teams use AI for each task, the share rating it very or extremely effective rose:

- Adding or updating functionality in existing code: 42% to 68%
- Refactoring or optimizing existing code: 43% to 67%
- Assisting development of new code: 55% to 79%
- Finding bugs: 44% to 63% (last year's wording: "debugging code")
- Code review: 47% to 62%

AI got much better at working with existing code

How effective are AI coding tools / agents for each of the following tasks you or your team / company has used them for? · % very or extremely effective, among teams using AI for each task



†Last year asked as "vibe coding (creating new projects with mostly AI-generated code)".

Respondents whose team uses AI for each task; bases vary by task · This year n=1,026 · Last year n=1,149 total · *Significant change vs. last year

FIG. 06

Code volume is rising, but many question whether the code is too verbose. 82% of developers and engineering managers agree they generate more lines of code than before AI, up from 64% last year. But 55% also agree “too many lines of code are being generated with AI,” up from 43%.

Code volume is rising, but many question whether the code is too verbose

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools and AI-assisted or generated code? | Choices: Completely agree / Somewhat agree



Last year n=1,149 · This year n=1,026 · *Significant change vs. last year

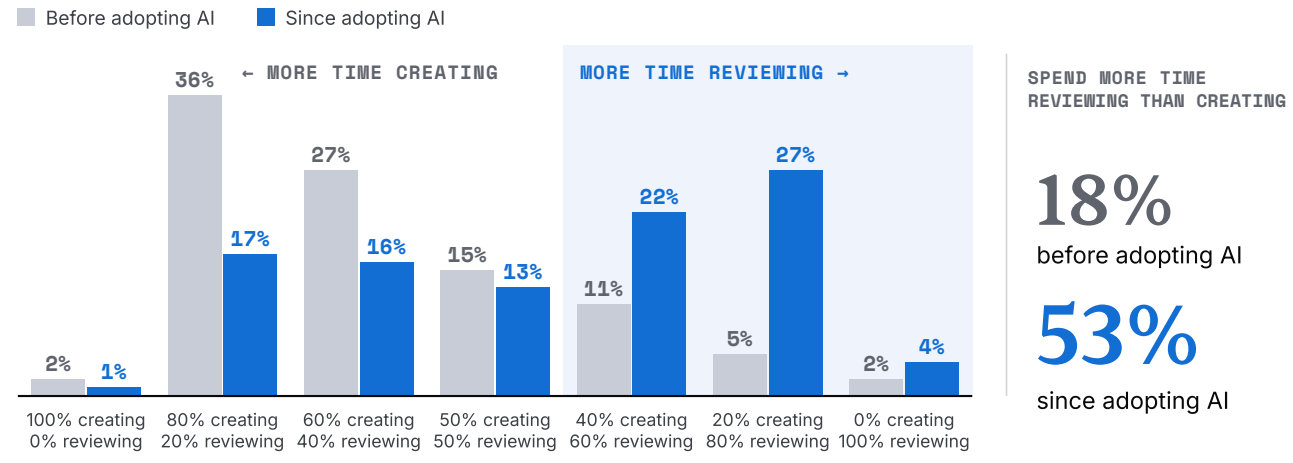
FIG. 07

Review now fills the week

53% of respondents now spend more time reviewing code than creating it, compared with 18% who say that was true before they adopted AI tools. And as time spent reviewing code has grown, the share who mostly create code fell from 67% to 34%.

Review has taken over the working week

How is your working time split between creating code and reviewing code... ...before adopting AI tools? | ...since adopting AI tools?



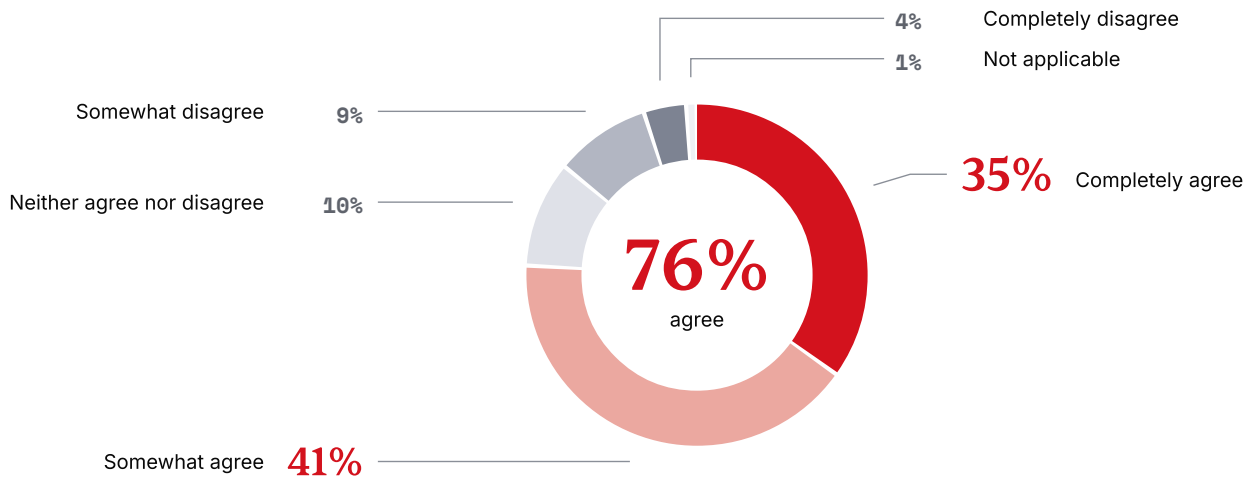
n=1,026 · “Don't know / not sure” (1% before, 1% since) not shown

FIG. 08

Despite the additional time spent reviewing code, 76% of developers and engineering managers (or roughly 3 out of 4) agree “we generate AI code faster than we can verify it.” Only 4% are comfortable enough with their review backlog to completely disagree with this statement.

AI code is being generated faster than it can be verified

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools and AI-assisted or generated code? | Choice: We generate AI code faster than we can verify it



n=1,026

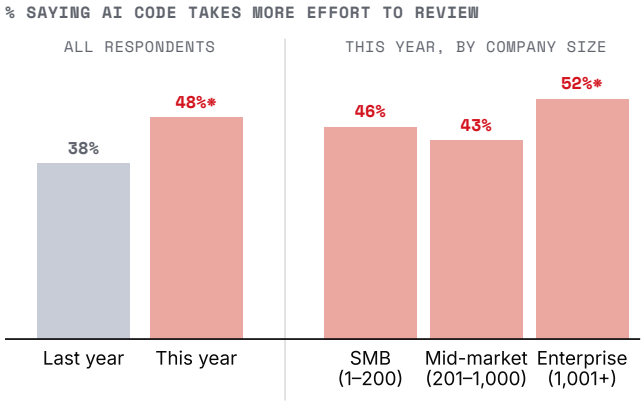
FIG. 09

For many, code review is also getting more challenging. 48% of respondents say reviewing AI-generated code requires somewhat or much more effort than reviewing code written by developers, up from 38% last year. This burden lands hardest at scale, where 52% of enterprise respondents say AI code takes more effort to review, vs. 43% at mid-market companies. Additionally, 49% agree reviewing AI-generated code is more time-consuming than reviewing developer-written code, up from 37% last year.

Each review is also getting heavier

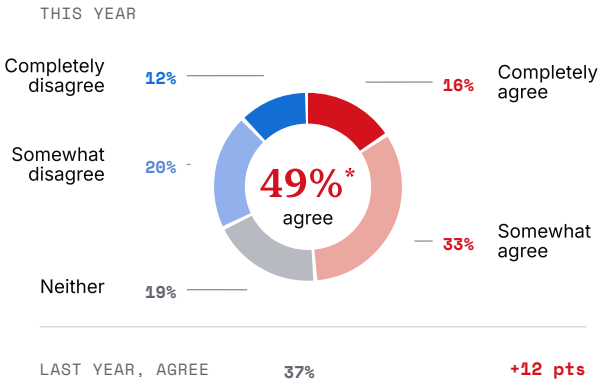
Reviewing AI code takes more effort

Does reviewing AI generated code require more or less effort than reviewing code written by developers? Reviewing AI generated code requires...



Reviewing AI code takes more time

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools and AI-assisted or generated code?
Choice: Reviewing AI-generated code is more time consuming than developer-written code



Excl. don't know · Last year n=1,138–1,149 · This year n=1,018–1,026 (SMB 282, mid 269, ent. 467) · *Significant vs. last year, or vs. mid-market · N/A (<1%) not labeled

FIG. 10

> "Developers are increasingly acting as editors and code reviewers rather than authors."
>
> SOFTWARE ENGINEER · ASIA-PACIFIC · MID-MARKET · TECHNOLOGY

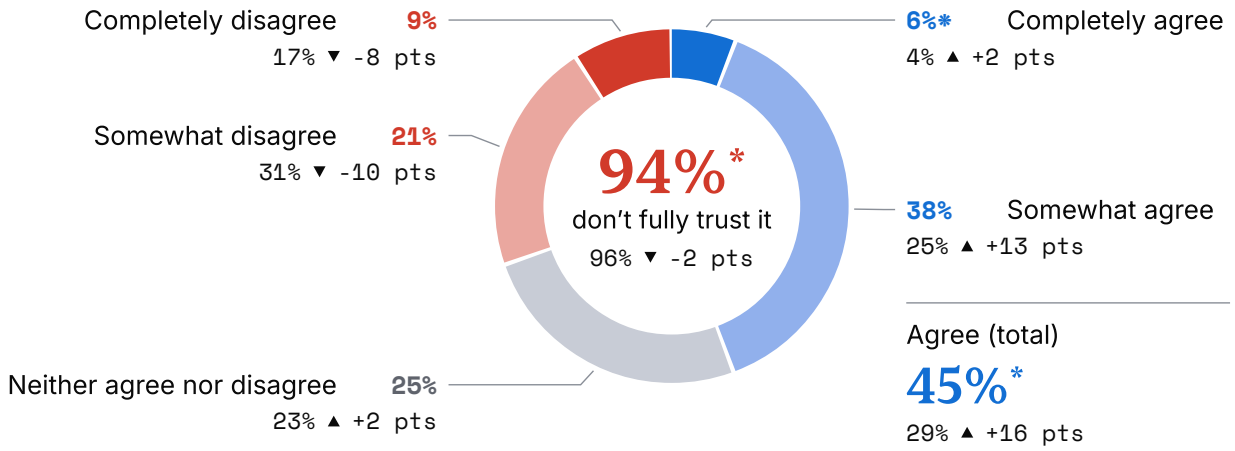
Trust is deepening, but it is not enough

Trust in AI code deepened this year, and consistent checking of the code fell.

45% of developers and engineering managers agree AI-generated code is functionally correct, up from 29%. Only 6% agree completely (4% last year), but 94% still do not fully trust it.

Trust in AI code is rising, but few trust it fully

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools and AI-assisted or generated code? | Choice: I trust that AI-generated code is functionally correct



Last year n=1,149 · This year n=1,026 · *Significant change vs. last year

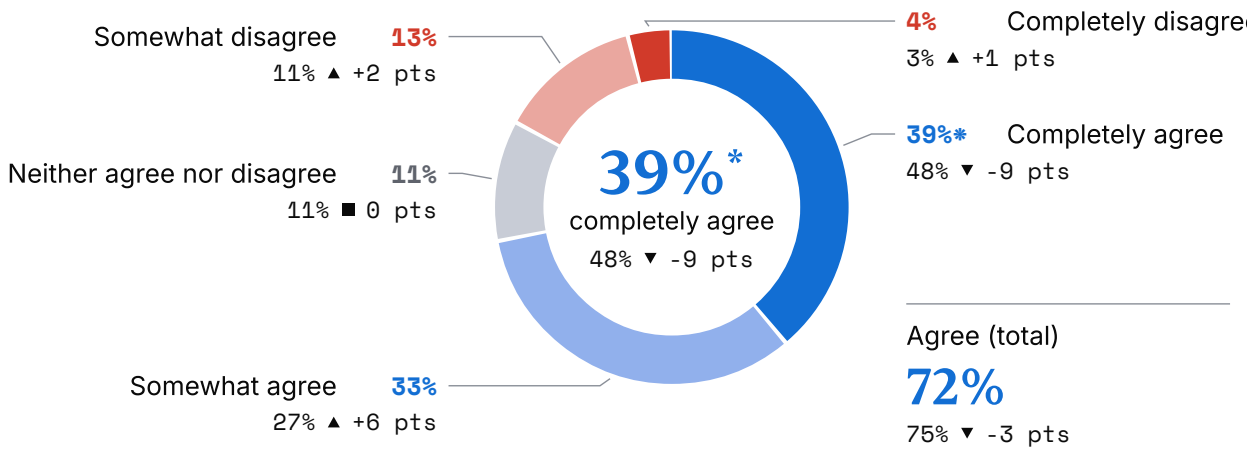
Black = last year's value · change in points: ▲ increase ▼ decrease ■ no change

FIG. 11

At the same time, the share who completely agree they always check AI-generated code before committing fell from 48% to 39%. Put together, this is consistent with what researchers call cognitive surrender, where people accept AI output with little scrutiny and let it replace their own judgment, often without realizing they have handed the decision over.

Fewer people check every change

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools and AI-assisted or generated code? | Choice: I always check my AI-generated or assisted code before committing it



Last year n=1,149 · This year n=1,026 · *Significant change vs. last year

Black = last year's value · change in points: ▲ increase ▼ decrease ■ no change

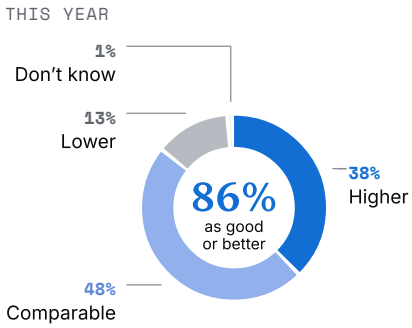
FIG. 12

But better-looking code can make problems harder to see. 86% now rate AI-generated code as comparable to (48%) or higher quality than (38%) code they write by hand. Agreement that "AI often produces code that looks correct but isn't reliable" fell from 61% to 49%. Meanwhile, concern that AI code is so well written that bugs are harder to spot rose from 31% to 42%.

Better-looking code makes problems harder to see

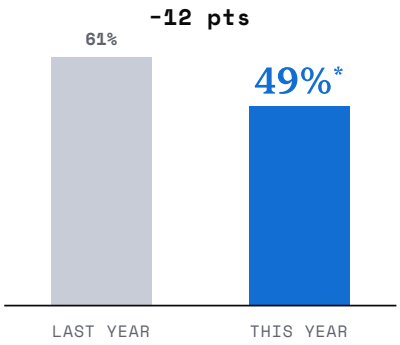
AI code quality vs. hand-written

How does the quality of AI-generated code compare to the code written by you or your team?



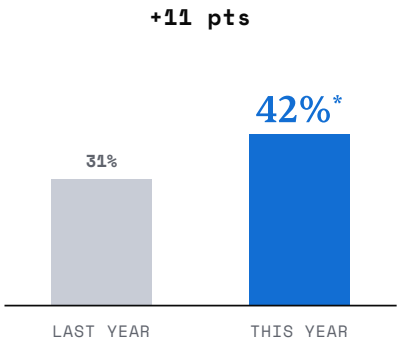
"Looks correct but isn't reliable"

To what extent do you agree...?
Choice: AI often produces code that looks correct but isn't reliable
Answer: Completely / somewhat agree



Polished code hides bugs

How concerned are you about...?
Choice: AI-generated code looking so well written that it becomes harder to spot bugs or issues
Answer: Very / extremely concerned



Last year: n=1,149 · This year: n=1,026 · *Significant change vs. last year

FIG. 13

> **"[The] biggest challenge is ensuring contextual consistency and preventing subtle, hard to spot security vulnerabilities or architecture drift that arise when developer[s] accept functional code [generated] without deep [awareness]."**
>
> SENIOR SOFTWARE ENGINEER · ASIA-PACIFIC · MID-MARKET · TECHNOLOGY

THE TAKEAWAY

Code is abundant. AI writes most of it, in the systems that matter most, and developers rate it better than ever. Confidence is scarce because verification has not scaled with generation, review fills the week, and as AI code gets more convincing, fewer people check every change.

But teams are responding by giving agents more autonomy, not less. The next chapter looks at where that leads.

CHAPTER 2

The software factory is coming fast

5% create or use software factories today. 34% expect to within a year.

Most teams have moved past experimenting with AI. 85% of respondents use company-approved coding agents, and 5% have even built software factories where agents work autonomously.

This chapter introduces a five-stage adoption journey for AI coding, shows where respondents sit today and where they expect to move in a year, and looks at what changes on the way.

Five stages of AI adoption

Which of the following best represents you / your team's current level of AI adoption?

STAGE 1	STAGE 2	STAGE 3	STAGE 4	STAGE 5
Limited or unsanctioned	**Piloting**	**Approved agents, few controls**	**Governed agents**	**Software factories**
AI use is limited or unsanctioned; some developers may use personal tools	There is some piloting of AI coding tools and approaches	There are approved coding agents available to use, but few controls or common standards	Coding agents are available to developers, and there are controls on token usage, permissions, coding standards, and tools	We are creating / using software factories (also known as "dark factories") where agents can work autonomously on tasks
3% of respondents	**12%** of respondents	**28%** of respondents	**52%** of respondents	**5%** of respondents

n=1,026

FIG. 14

WHAT IS A SOFTWARE FACTORY?

In this survey, a "software factory" is the fifth and most advanced stage of AI adoption, described to respondents as: "We are creating/using software factories (also known as 'dark factories') where agents can work autonomously on tasks." The idea borrows from manufacturing. Modern factories build quality into every step, so problems are caught where they start. A software factory applies the same discipline to software development: it standardizes the process while allowing each cycle to produce a different solution. In a software factory, agents carry work through increasingly autonomous cycles of planning, implementation, testing, and refinement. People provide the goals, context, quality standards, and feedback that serve as guiderails throughout those cycles. Because the survey used the term "dark factories," Stage 5 leans toward fully autonomous setups with little human review.

Most teams already run agents, and half govern them

52% of respondents say their teams are operating in Stage 4, having coding agents available with controls on token usage, permissions, coding standards, and tool access. Another 28% sit at Stage 3, with approved coding agents but few controls or common standards. 12% are piloting (Stage 2), 3% describe limited or unsanctioned use (Stage 1), and 5% are creating or using software factories (Stage 5). But respondents expect the software factory to grow to 34%, over sevenfold, in a year.

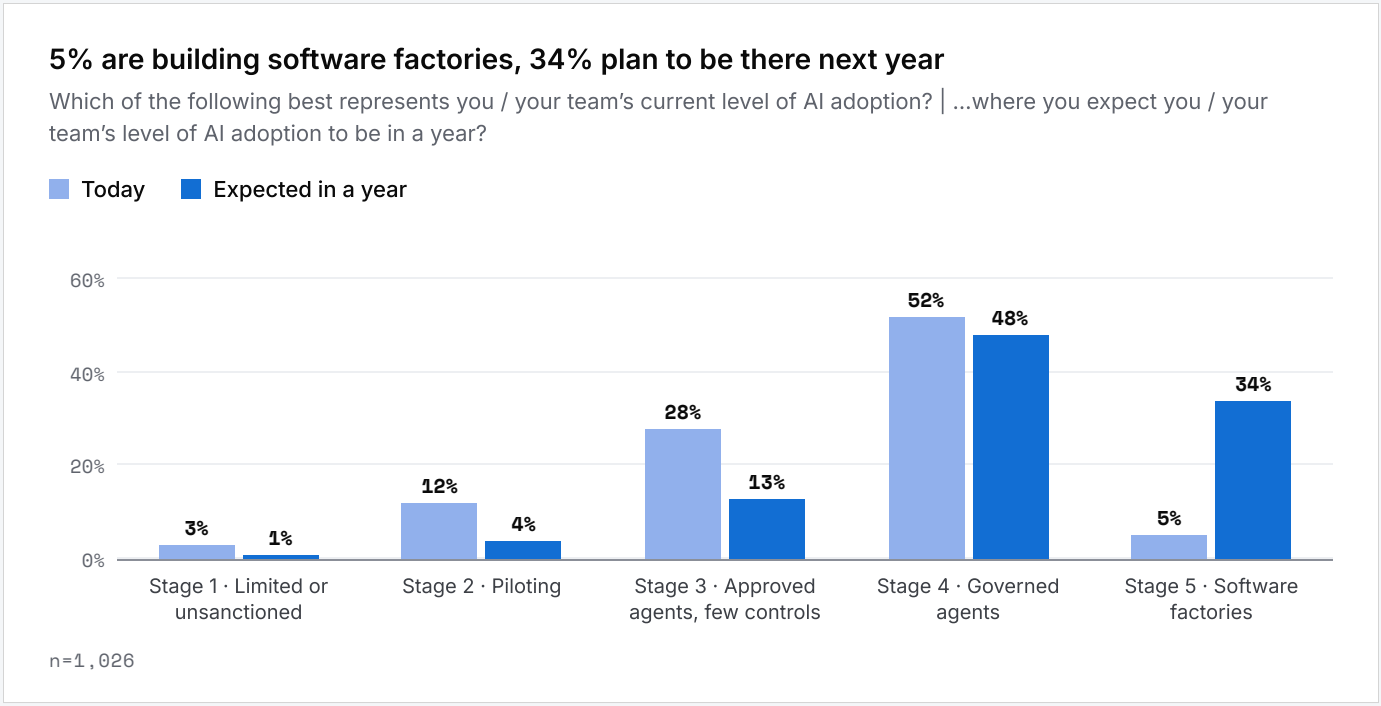


FIG. 15

Within the next year, 62% of respondents expect their team to move up at least one stage, and 11% expect to skip at least one. If those expectations play out even partly, a large majority of teams are poised to hand agents significantly more autonomy.

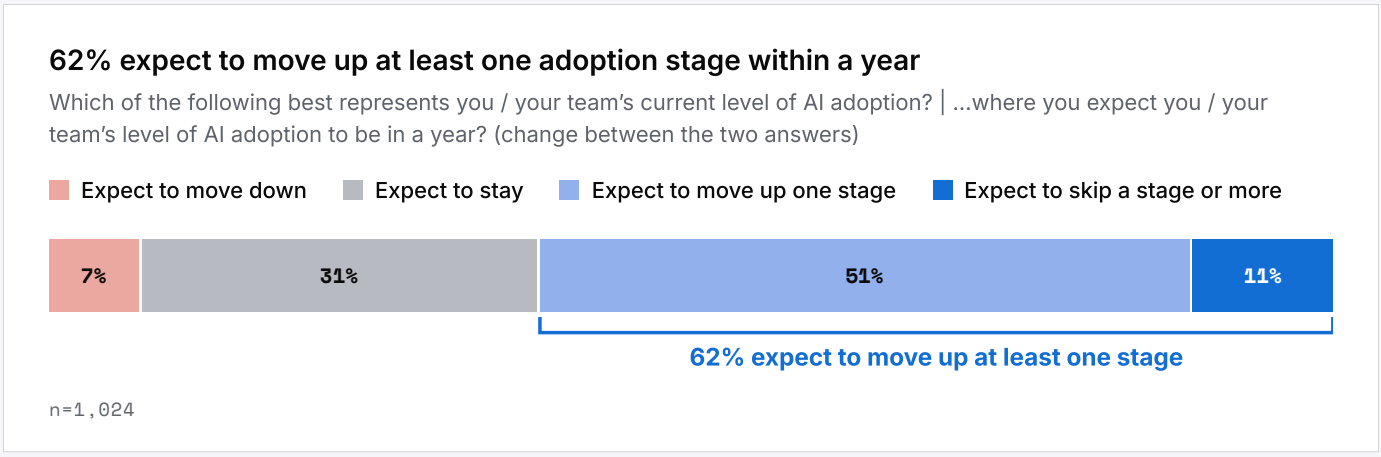


FIG. 16

Tech and financial services lead the way

Leading the pack are 40% of respondents in technology and 39% in financial services, who expect to be creating or using software factories within a year.

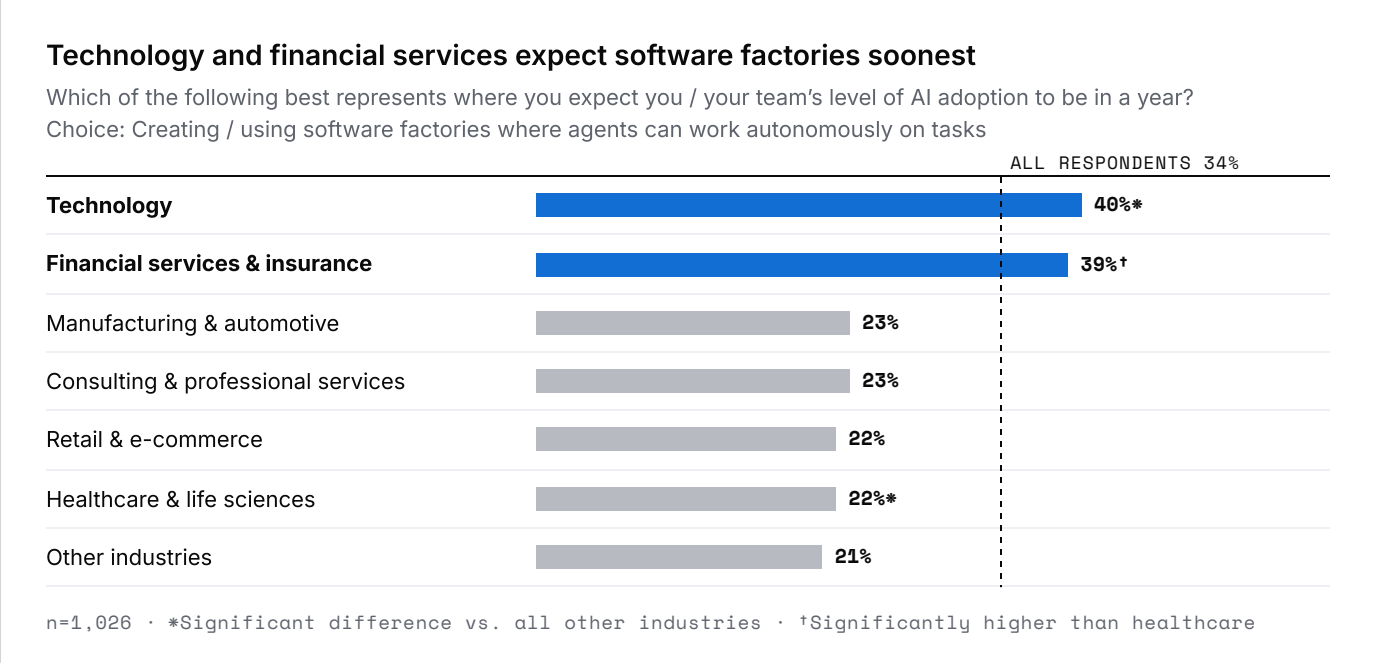


FIG. 17

Financial services stands out as a regulated industry that is still among the most ambitious. By contrast, healthcare appears more cautious: only 22% are aiming for software factories next year.

Guiding agents: Context and harness usage varies widely

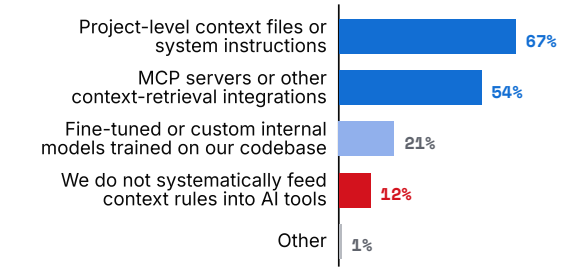
The mechanisms teams use to guide their agents can vary quite a bit. 67% of respondents say their team supplies codebase context or rules through project-level context files or system instructions, and 54% through Model Context Protocol (MCP) servers or other context-retrieval integrations. 12% do not systematically feed architecture or context rules into AI tools at all.

Application of harnesses is also not consistent. 45% of respondents either let individuals pick their own harness (24%) or use whatever ships with the model (21%). That is likely to produce inconsistent output across a team, because the same model behaves differently depending on how it is wired to the codebase and tools.

Centralized context supply is on the rise, but harness selection is still up to individuals

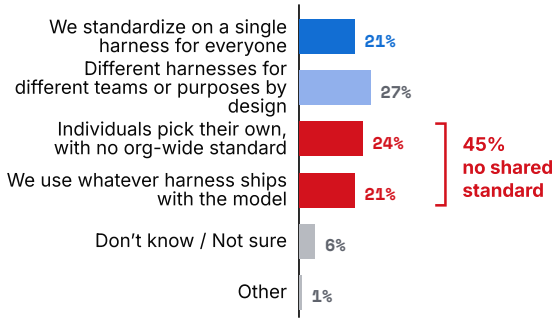
How teams supply context to agents

How does your team currently supply codebase context, architectural rules, or quality guardrails to AI coding tools / agents? (select all that apply)



How organizations pick agent harnesses

Which of the following best describes your organization's approach to selecting and standardizing coding agent harnesses?



n=1,026

FIG. 18

> **"Teams need better ways to preserve system context... and make AI usage auditable without slowing everyone down."**
>
> ENGINEERING MANAGER · NORTH AMERICA · ENTERPRISE · TECHNOLOGY

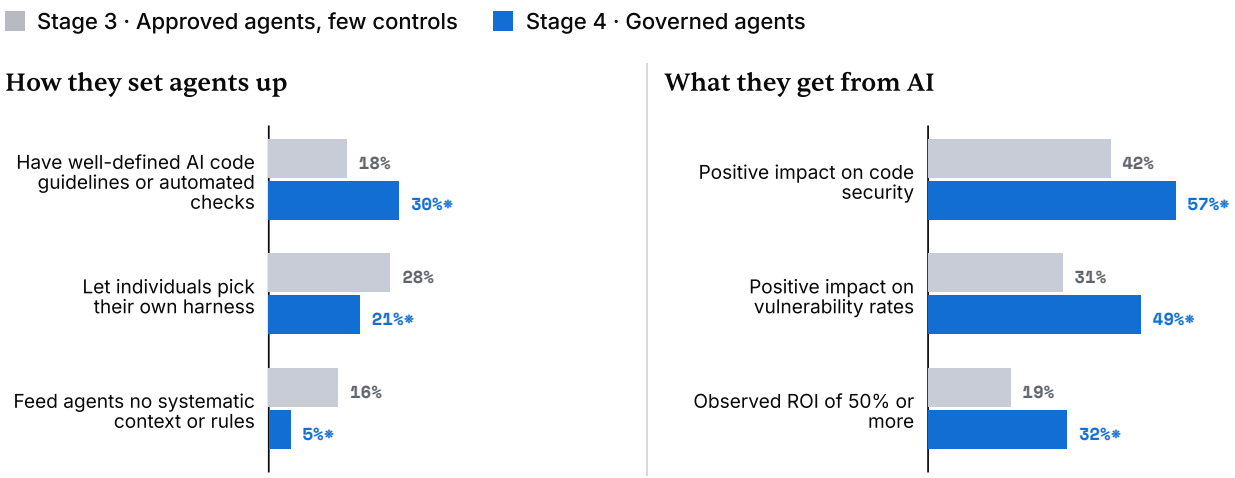
Teams that put controls on their agents report better security and ROI

Stage 3 is where teams have given developers capable agents but have not yet written down the standards, context, and controls those agents need. Compared with Stage 4, Stage 3 teams are less likely to have well-defined AI guidelines (18% vs. 30%), more likely to let individuals pick their own harness (28% vs. 21%), and three times as likely to feed agents no systematic context (16% vs. 5%).

These teams report the weakest security results of any stage, and teams using governed agents (Stage 4) report stronger security results.

- Code security: 57% of Stage 4 respondents say AI has had a positive impact, vs. 42% at Stage 3.
- Vulnerability rates: 49% of Stage 4 respondents report a positive impact on vulnerability rates, vs. 31% at Stage 3.
- Returns: 32% of Stage 4 respondents report an observed ROI of 50% or more, vs. 19% at Stage 3.
- Business outcomes: Stage 4 respondents report a positive impact significantly more often than Stage 3 on all 14 business outcomes we measured.

Governed agents pull ahead of teams with few controls



Stage 3 n=290 · Stage 4 n=535 · *Significant difference vs. Stage 3

FIG. 19

> **"The challenge is to define the correct constraints for the AI to operate autonomously and effectively and with correctness."**
>
> SOFTWARE DEVELOPMENT ENGINEER · LATIN AMERICA · ENTERPRISE · RETAIL

THE TAKEAWAY

Most teams are already closing in on an agentic future, and a third expect to have software factories within a year. The stage in the middle of the climb, where agents are used without guiderails, is where risk concentrates. Teams with governed agents report stronger security and higher returns.

The more autonomy agents get, the higher the stakes when something slips through. The next chapter looks at the stakes developers worry about most: security.

CHAPTER 3

Security concerns dominate, even as more teams say AI reduced outage frequency

Respondents' top security concern is protecting their code from AI-powered attackers

Last year, developers told us their biggest worry about AI was what it might leak: exposure of sensitive company or customer data. This year we added a new item, and it went straight to the top. 61% of developers and engineering managers are very or extremely concerned about attacks that use AI to find and exploit vulnerabilities faster and at greater scale.

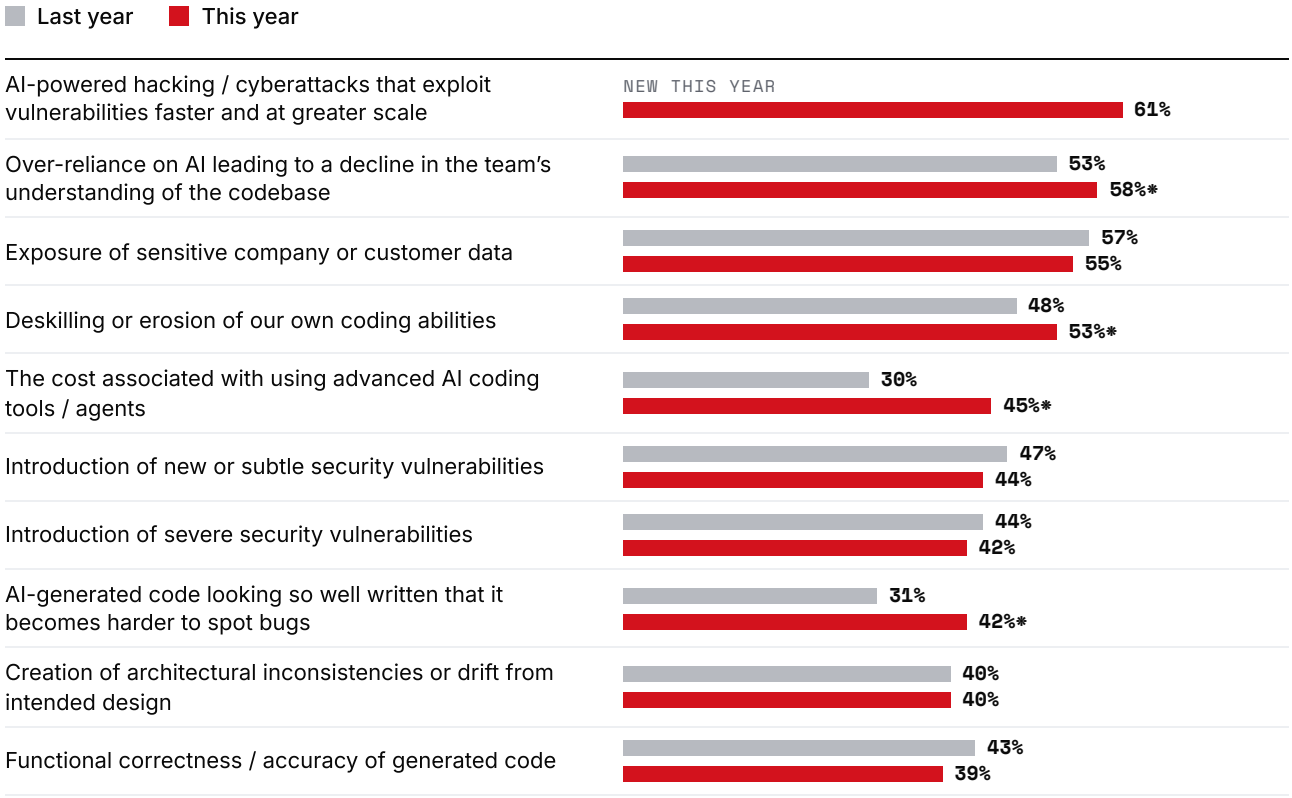
That is a shift in how developers see the threat. The risk is not only that AI writes flawed code. Attackers now have AI too, and AI has been shown to excel at finding latent vulnerabilities in code that may not have been touched for years.

Security dominates the list of AI concerns

These are the 10 issues with AI-generated code that the most respondents say they are very or extremely concerned about.

AI-powered attacks are the #1 AI concern

How concerned are you about each of the following when it comes to AI generated code today? % very or extremely concerned · top 10 of 27 concerns this year



This year n=1,026 · Last year n=1,149 · *Significant change vs. last year

FIG. 20

More broadly, four of the top seven concerns are about security. In addition to AI-powered hacking, 44% are very or extremely concerned about new or subtle vulnerabilities and 42% about severe ones.

That changes the verification problem. If attackers can find a flaw within hours of it shipping, catching it in a quarterly security review is too late.

> **"There is bound to be a miss, or a zero-day vulnerability introduced by such code. Who takes accountability for it is still unclear."**
>
> SENIOR PRODUCT MANAGER · NORTH AMERICA · ENTERPRISE · TECHNOLOGY

AI is helping security more than hurting it

When we asked respondents about the outcomes they are seeing right now, the picture gets better:

- 52% say AI-generated code has had a positive impact on code security, vs. 14% negative.
- 28% say AI reduced security vulnerabilities, vs. 15% who say it increased them.
- 36% say AI had a positive effect on the frequency of security breaches or data leaks, vs. 10% negative.

Shares do not add up to 100% because the rest of respondents reported neither a positive nor a negative impact.



FIG. 21

On every security measure, gains outnumber setbacks by about two to one or more. Looking year-over-year, the share crediting AI with fewer vulnerabilities rose 10 points, while the share blaming it for more stayed flat.

AI is also emerging as a valuable tool for fixing security issues, with 61% of respondents' teams using AI for vulnerability patching or remediation. But only 56% rate it very or extremely effective, below most other tasks.

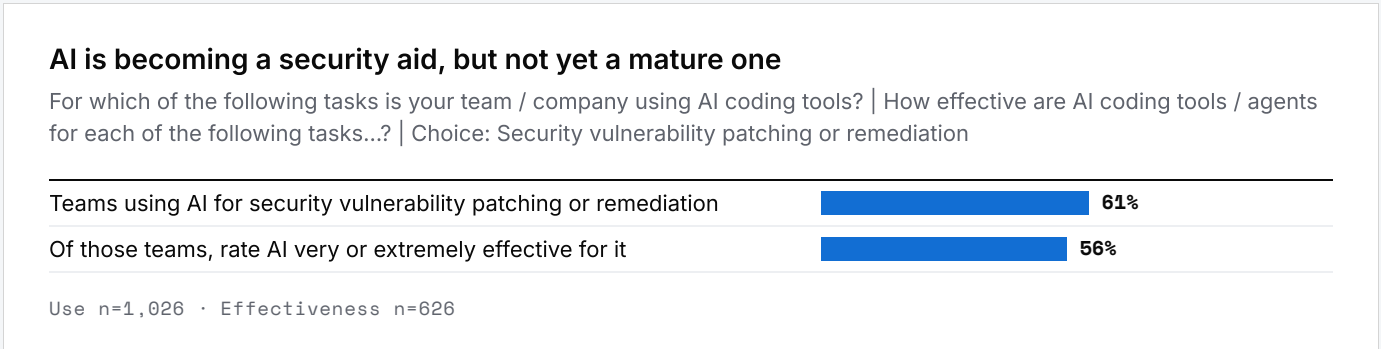


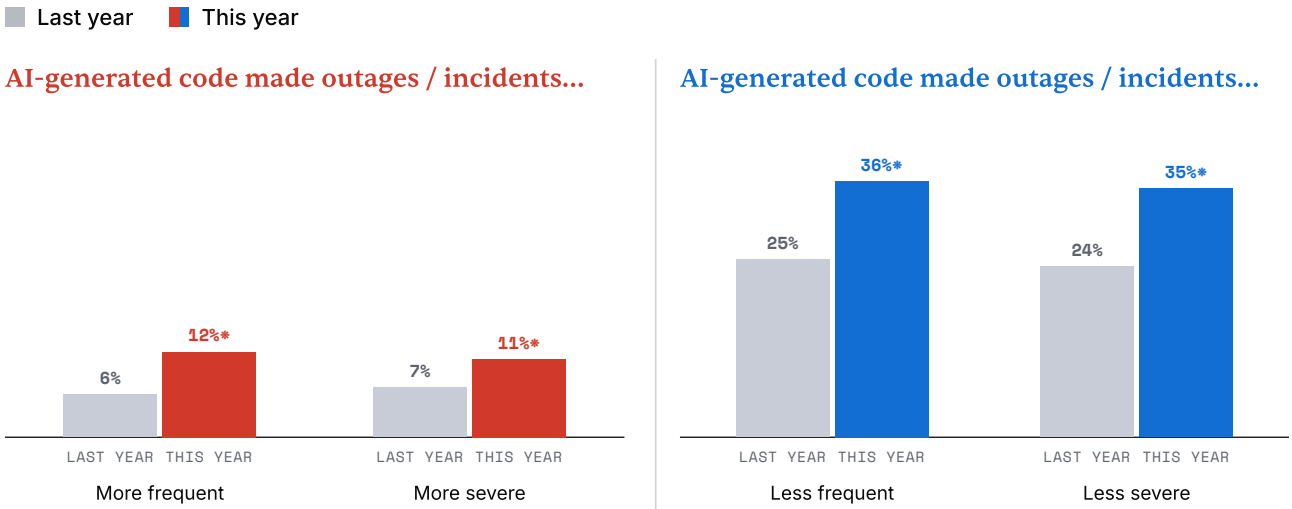
FIG. 22

More teams say AI reduced outages vs. caused them

12% say AI-generated code made outages or incidents more frequent, nearly double last year's 6%. 11% say it made them more severe, up from 7%. But, interestingly, that does not mean AI is making systems less reliable overall. 36% say AI reduced outage frequency, up from 25%, and 35% say AI reduced outage severity, up from 24%.

Reliability setbacks nearly doubled, but the gains grew, too

What impact has AI-generated code had on your team / company for each of the following? | Choices: Frequency of outages / incidents; Severity of outages / incidents



Last year n=1,149 · This year n=1,026 · *Significant change vs. last year

FIG. 23

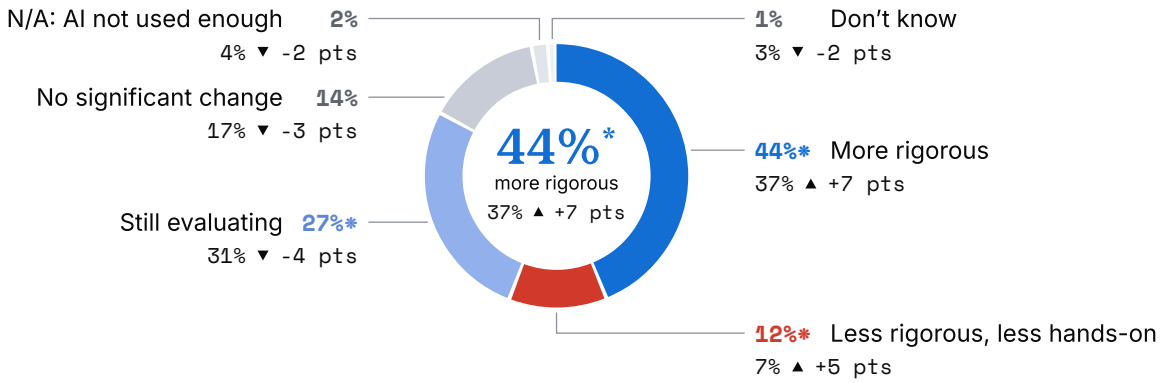
> **"AI can generate so much so quickly that it can't all be reviewed. AI has shown it can find and exploit vulnerabilities faster than people can find and fix them."**
>
> SENIOR SOFTWARE ENGINEER · NORTH AMERICA · MID-MARKET · FINANCIAL SERVICES

Rigor is rising and splitting

Teams are responding, but not all in the same way. 44% say they are more rigorous about code security because of AI-generated code, up from 37%. At the same time, 12% say they are less rigorous and less hands-on, up from 7%. Fewer teams are still deciding their way forward (31% to 27%).

Teams are responding on security, but not in the same direction

Has there been a change in how you approach the following as a result of the increasing use of AI coding tools / agents? | Choice: Code security



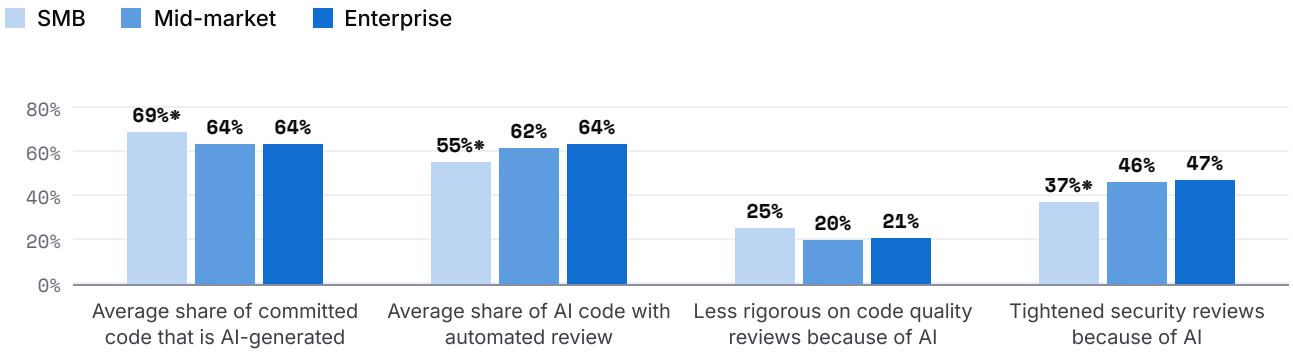
Last year n=1,149 · This year n=1,026 · *Significant change vs. last year
Black = last year's value · change in points: ▲ increase ▼ decrease ■ no change

FIG. 24

Company size makes a modest but consistent difference. At small and midsize businesses (SMBs), an average of 69% of committed code is AI-generated or assisted, vs. 64% at enterprises. Yet SMBs run automated reviews on only 55% of their AI code (vs. 64% for enterprises), and 25% say they have become less rigorous on code quality reviews because of AI (vs. 21% for enterprises). 47% of enterprise respondents have tightened security reviews, vs. 37% of SMBs. None of these gaps is large on its own, but they all suggest smaller teams write a bit more of their code with AI and are less likely to have raised the bar on reviewing it.

Smaller teams write more code with AI, and review less of it

What percentage of the code you personally committed was AI-generated or assisted? | What percentage of AI code is reviewed...? | Has there been a change in how you approach the following as a result of the increasing use of AI coding tools?



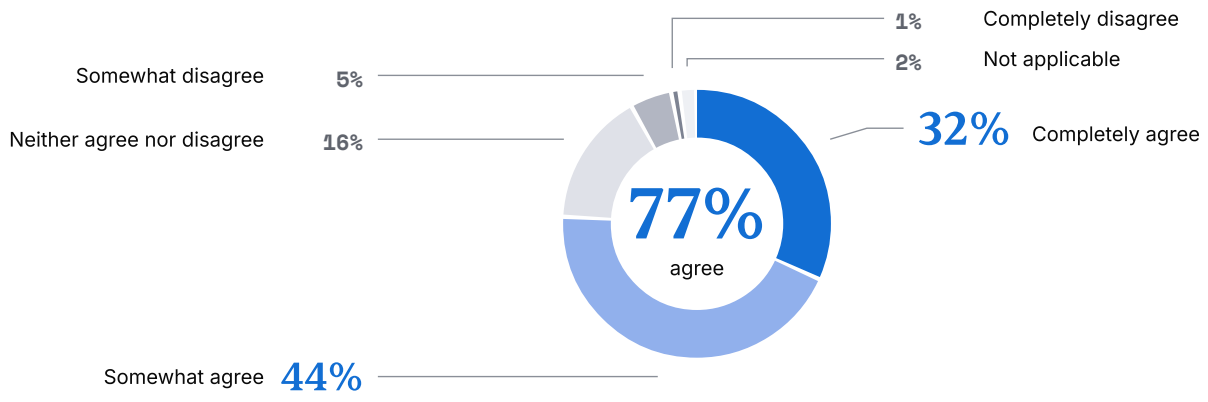
SMB (1-200) n=286 · Mid-market (201-1,000) n=271 · Enterprise (1,001+) n=469 ·
AI share of code among respondents who write code: n=246 / 222 / 388 ·
*Significant difference, SMB vs. enterprise

FIG. 25

Respondents also tell us what would raise their confidence. 77% of developers and engineering managers agree that strong code verification tools make them, or their team, trust AI code more, with 32% of them agreeing completely.

Strong verification tools raise trust in AI code

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools / agents? | Choice: Having strong code verification tools makes me / our team trust AI code more



n=1,026

FIG. 26

> **“The biggest challenge is trust and validation: AI-generated code can introduce subtle security vulnerabilities or poor design, so strong human review, automated testing, and security checks are essential.”**
>
> SENIOR SOFTWARE ENGINEER · ASIA-PACIFIC · SMB · TECHNOLOGY

THE TAKEAWAY

Developers’ security worries have moved outward, toward AI-powered attackers, while their own experience with AI-generated code is improving on balance. Reliability has also seen impacts from AI coding, but in both directions. The share of respondents reporting increased outage frequency has doubled, while three times as many respondents say AI reduced outage frequency (36% vs. 12%).

Security concern is pushing organizations to formalize how AI code is governed. The next chapter looks at how far they have gotten.

CHAPTER 4

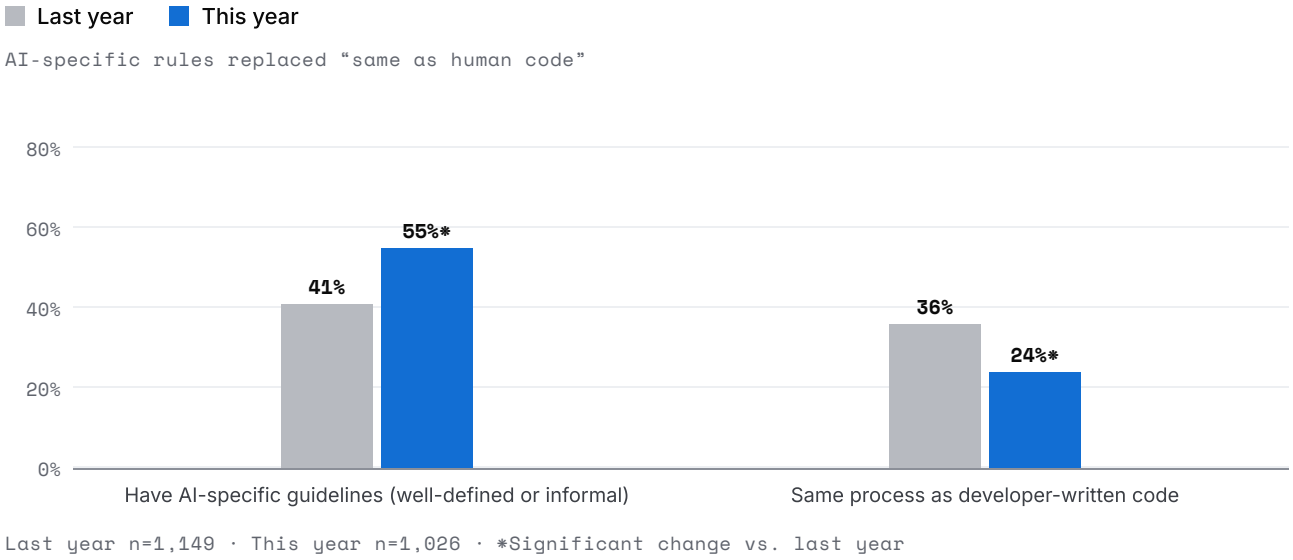
Governance is spreading, but few teams define it well

Most teams now have rules for AI code. Fewer than one in four have rules they can enforce.

A year ago, 41% of respondents reported that their organization had guidelines, policies, or quality gates for AI-generated code that differ from those for developer-written code. Now, that figure has risen to 55%. In the same period, respondents reporting using the same guidelines for both AI- and developer-generated code dropped from 36% to 24%.

Most teams now have rules for AI code

Does your organization currently have specific guidelines, policies, or quality gates for the review, testing, and acceptance of AI-generated code that differ from those for developer-written code?



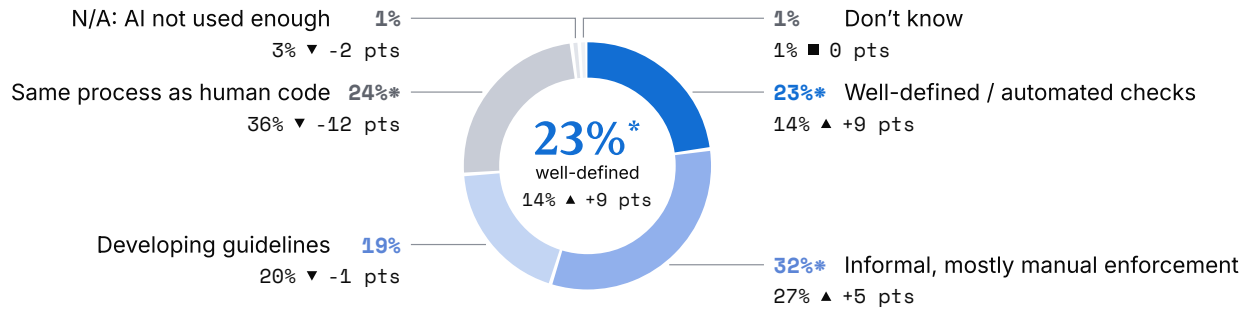
Last year n=1,149 · This year n=1,026 · *Significant change vs. last year

FIG. 27

But having well-defined rules an agent can be checked against is less assured. Only 23%, or less than a quarter, describe well-defined, distinct guidelines or automated checks (up from 14% last year). The most common approach is still informal rules enforced by hand (32%, up from 27% last year).

Well-defined rules grew fastest, but informal rules still lead

Does your organization currently have specific guidelines, policies, or quality gates for the review, testing, and acceptance of AI-generated code that differ from those for developer-written code?



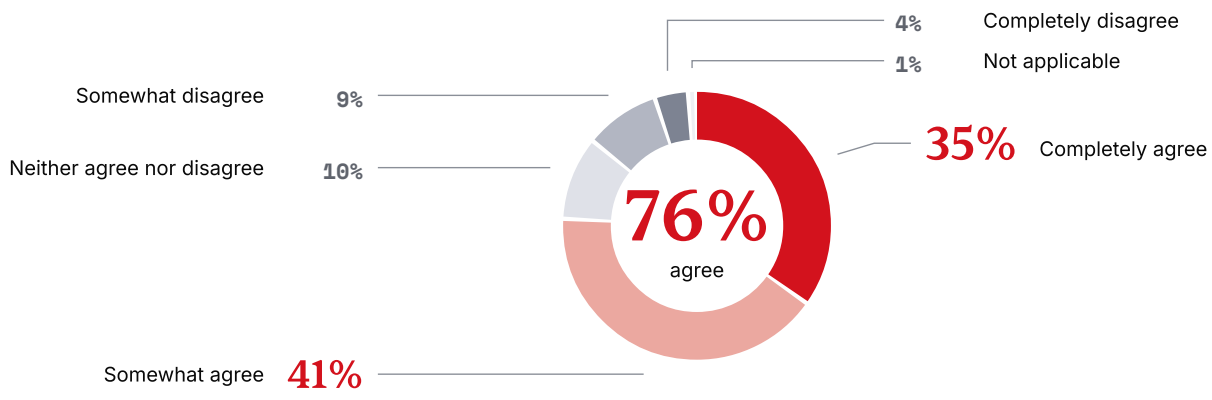
Last year n=1,149 · This year n=1,026 · *Significant change vs. last year
Black = last year's value · change in points: ▲ increase ▼ decrease ■ no change

FIG. 28

That matters because informal rules are enforced by people, and code review is where teams now run out of capacity. 76% of respondents report generating code faster than they can verify it.

AI code is being generated faster than it can be verified

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools and AI-assisted or generated code? | Choice: We generate AI code faster than we can verify it



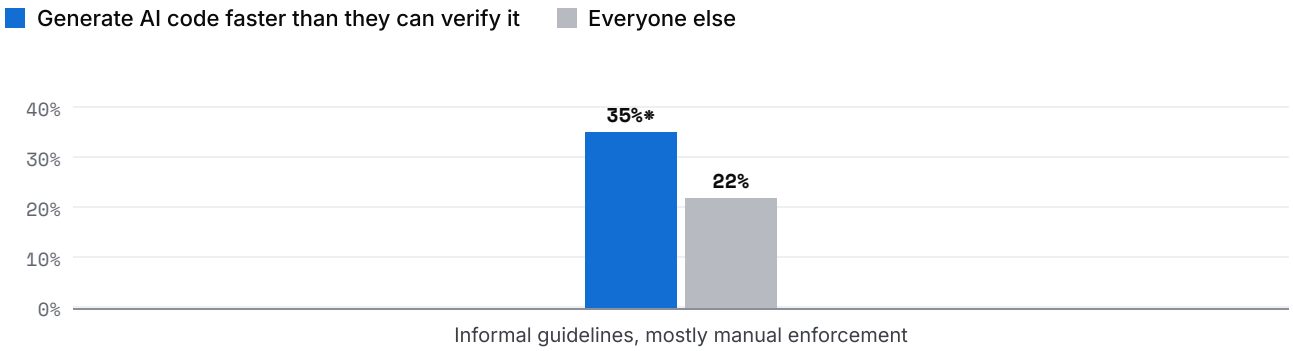
n=1,026

FIG. 29

The teams under the most pressure are leaning hardest on informal rules. Among respondents who say they generate AI code faster than they can verify it, 35% rely on informal guidelines, vs. 22% of everyone else.

Teams under the most pressure lean hardest on informal rules

Does your organization have specific guidelines, policies, or quality gates for AI-generated code that differ from those for developer-written code? · by agreement with "We generate AI code faster than we can verify it"



Generate faster than they can verify: n=780 · Everyone else: n=246 ·
*Significant difference vs. everyone else

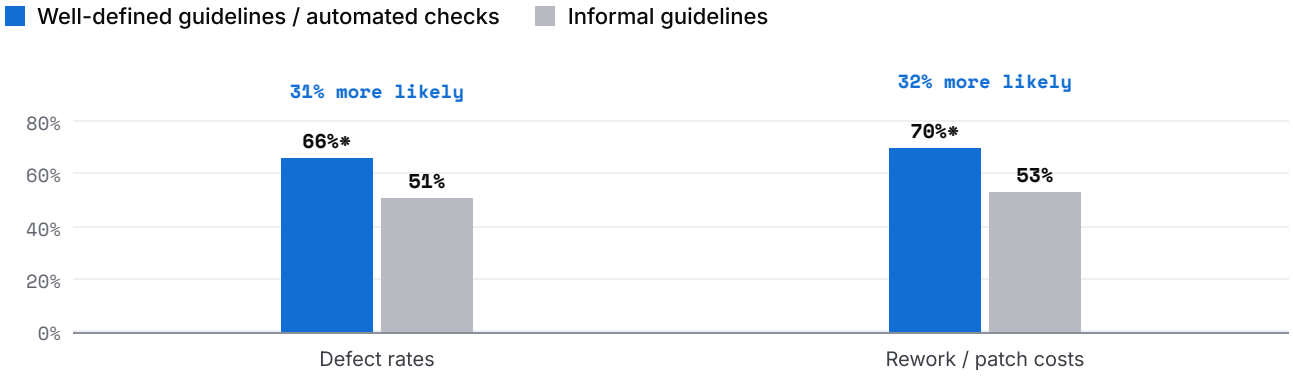
FIG. 30

Well-defined rules go with better quality and compliance outcomes

Teams with well-defined rules are 31% more likely to say AI has had a positive impact on defect rates (66% vs. 51%) and 32% more likely to say the same for rework or patch costs (70% vs. 53%).

Well-defined rules go with fewer defects and less rework

What impact has AI-generated code had on your team / company for each of the following? | Answer: Somewhat or very positive impact · by type of AI code guidelines



Well-defined n=239 · Informal n=330 · Relative differences from unrounded figures ·
*Significant difference vs. informal guidelines

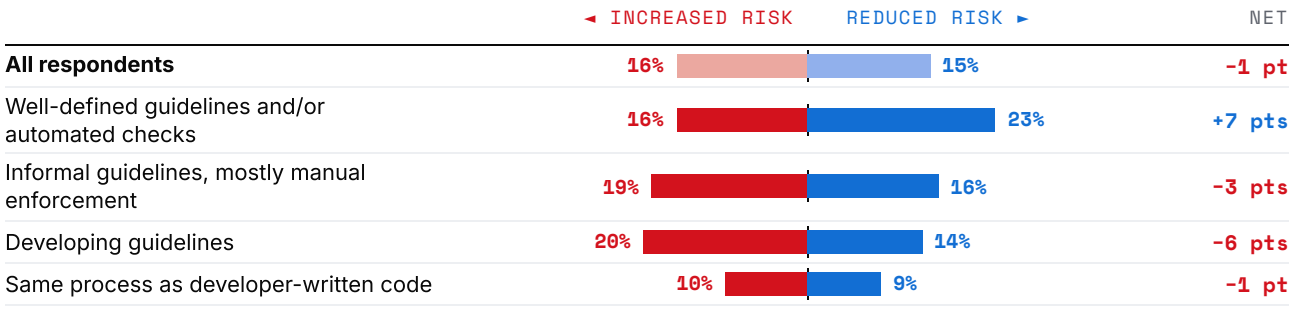
FIG. 31

Effects on compliance are more nuanced. AI cut compliance risk about as often as it raised it: 15% say AI-generated code reduced compliance or regulatory risk, and 16% say it increased it.

But the type of guidelines used changes the picture. Teams with well-defined guidelines are the only group where AI cut compliance risk more often than it raised it (23% reduced vs. 16% increased). With informal guidelines, the balance tips the other way (16% vs. 19%), and it tips further for teams still developing them (14% vs. 20%). Teams that treat AI code the same as human code report little change either way (9% vs. 10%).

Only well-defined guidelines see AI cut compliance risk more than raise it

In which of the following ways, if any, has AI-generated code impacted your team / company's technical debt? | Choices: Reduced compliance or regulatory risk; Increased compliance or regulatory risk · by AI code guidelines



All n=1,026 · Well-defined n=239 · Informal n=330 · Developing n=192 · Same as human n=248

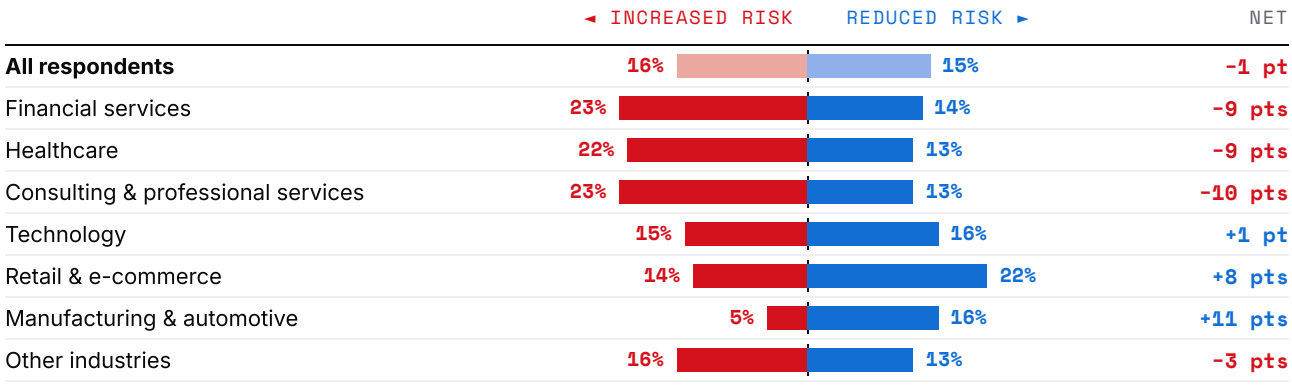
FIG. 32

> **"Code review guidelines and tools around code review will take [a] prominent position going forward to avoid security risks, [compliance] risks, etc."**
>
> HEAD OF DELIVERY · ASIA-PACIFIC · ENTERPRISE · TECHNOLOGY

Respondents are reporting more compliance risk where the rules are more strict. 23% of financial services respondents and 22% in healthcare say AI-generated code has increased their compliance or regulatory risk, compared with 16% overall, and 14% and 13% of respondents respectively who say these risks were reduced.

In regulated industries, AI raises compliance risk more often than it cuts it

In which of the following ways, if any, has AI-generated code impacted your team / company's technical debt? | Choices: Reduced compliance or regulatory risk; Increased compliance or regulatory risk · by industry



All n=1,026 · FinServ 106 · Healthcare 69 · Consulting 40 · Other 118 · Technology 601 · Retail 49 · Manufacturing 43

FIG. 33

Healthcare and financial services: Two regulated industries that approach governance differently

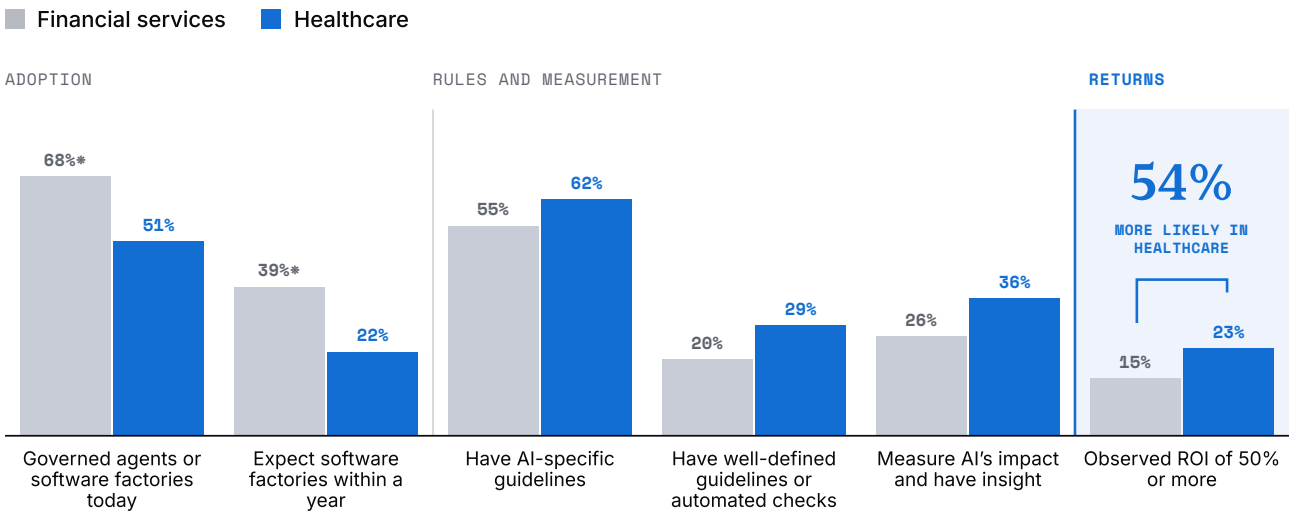
Financial services is charging ahead of healthcare in adopting governed agents, the stage from which most teams planning a software factory are moving up. 68% run governed agents or software factories, vs. 51% in healthcare, and 39% expect to run a software factory within a year (vs. 22%).

Yet more healthcare teams have well-defined rules for AI code (29% vs. 20% for financial services) or measure AI's impact (36% vs. 26%). Healthcare organizations have written down more of their rules, with 62% having AI-specific guidelines vs. 55% in financial services.

Healthcare respondents are also 54% more likely to report an ROI of 50% or more. The data does not show whether those practices explain the difference, but the pattern is worth watching.

Financial services moves faster, healthcare sees more gains

Regulated industries represented among survey respondents: financial services and healthcare · AI adoption stage, AI code guidelines, measurement of AI's impact and observed ROI



Financial services n=106 · Healthcare n=69 · 54% from unrounded figures (23.2% vs. 15.1%) · *Significant difference vs. healthcare

FIG. 34

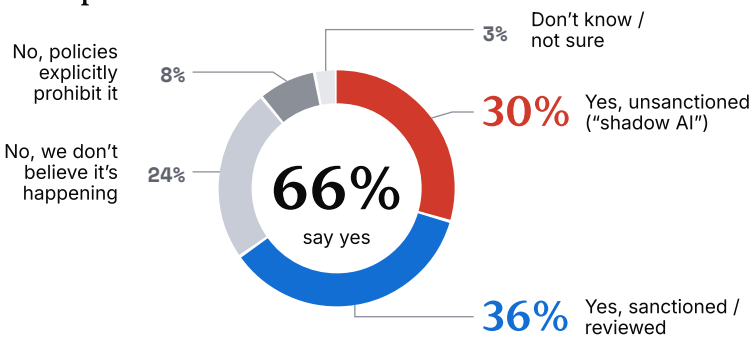
Software is being built outside engineering

Governance also has to reach people who are not engineers. 66% of respondents say non-engineers at their organization (product managers, designers, business analysts) use AI tools or agents to build or ship working software. For 36%, that is common and sanctioned or reviewed by IT or engineering. For 30%, it happens without official sanction. Interestingly, enterprises report more unsanctioned building than small and midsize businesses (SMBs), at 32% vs. 26%.

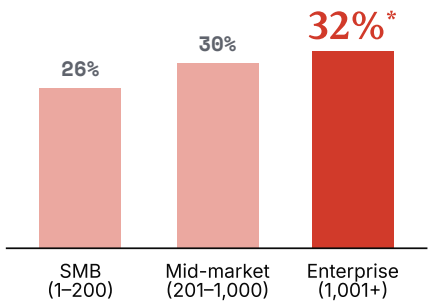
Two-thirds say non-engineers build software with AI, three in ten without sanction

Are non-engineers at your organization (e.g. product managers, designers, business analysts) using AI tools / agents to build or ship working software?

All respondents



Unsanctioned, by company size



All respondents n=1,026 · SMB n=286 · Mid-market n=271 · Enterprise n=469 · *Significant difference vs. SMB

FIG. 35

> **"Tooling has outpaced the governance around it. Most teams can generate code fast now, but very few have caught up on how to verify it properly at that speed."**
>
> DIRECTOR · EUROPE · ENTERPRISE · MEDIA & ENTERTAINMENT

THE TAKEAWAY

Governance has gone mainstream. The teams that get value from it turn policy into something specific enough for agents to follow and for automated checks to enforce. Without that, code piles up faster than the rules can catch it.

The teams under the most pressure are the most likely to rely on people to maintain coding standards. The next chapter looks at what that pressure does to the codebase.

CHAPTER 5

AI tends to make healthy codebases healthier and messy ones messier

AI is generally paying tech debt down, but there are exceptions

AI-generated code is reducing technical debt for the majority of teams. 56% of respondents say AI has had a positive impact on their team's technical debt overall, up from 47% last year, against only 16% who say the impact was negative.

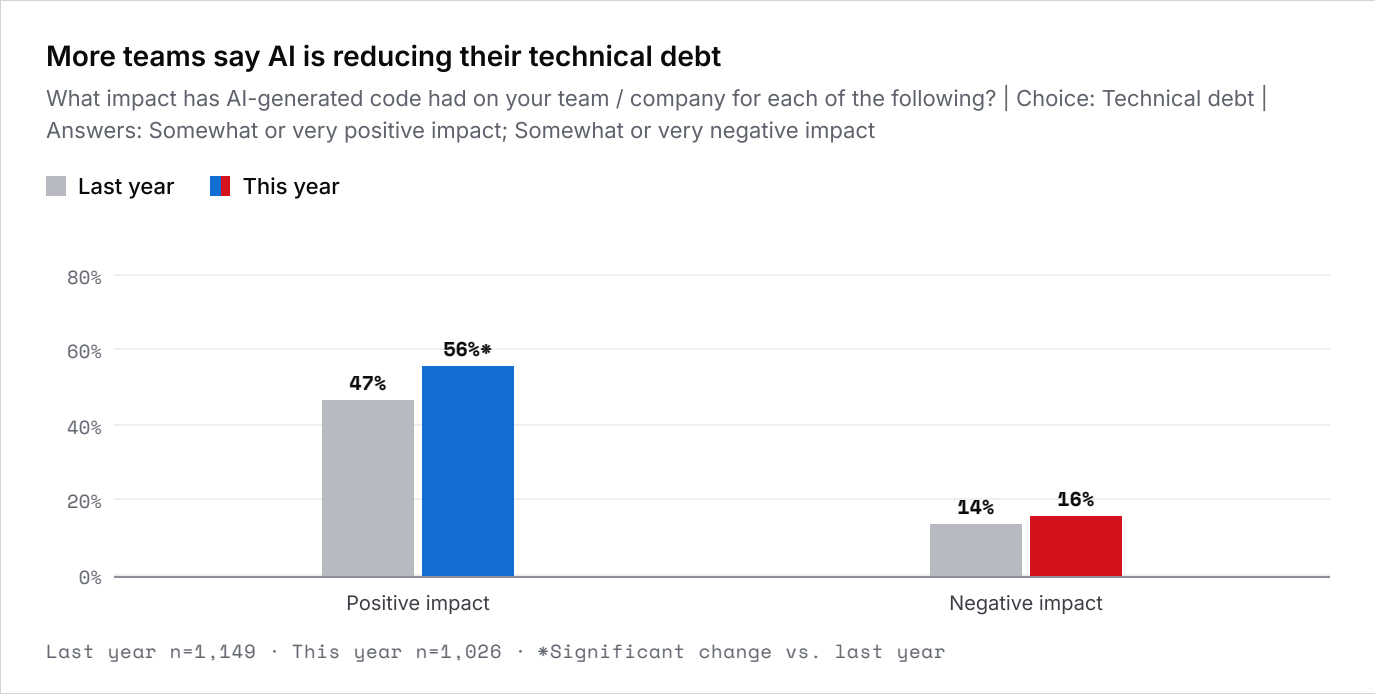
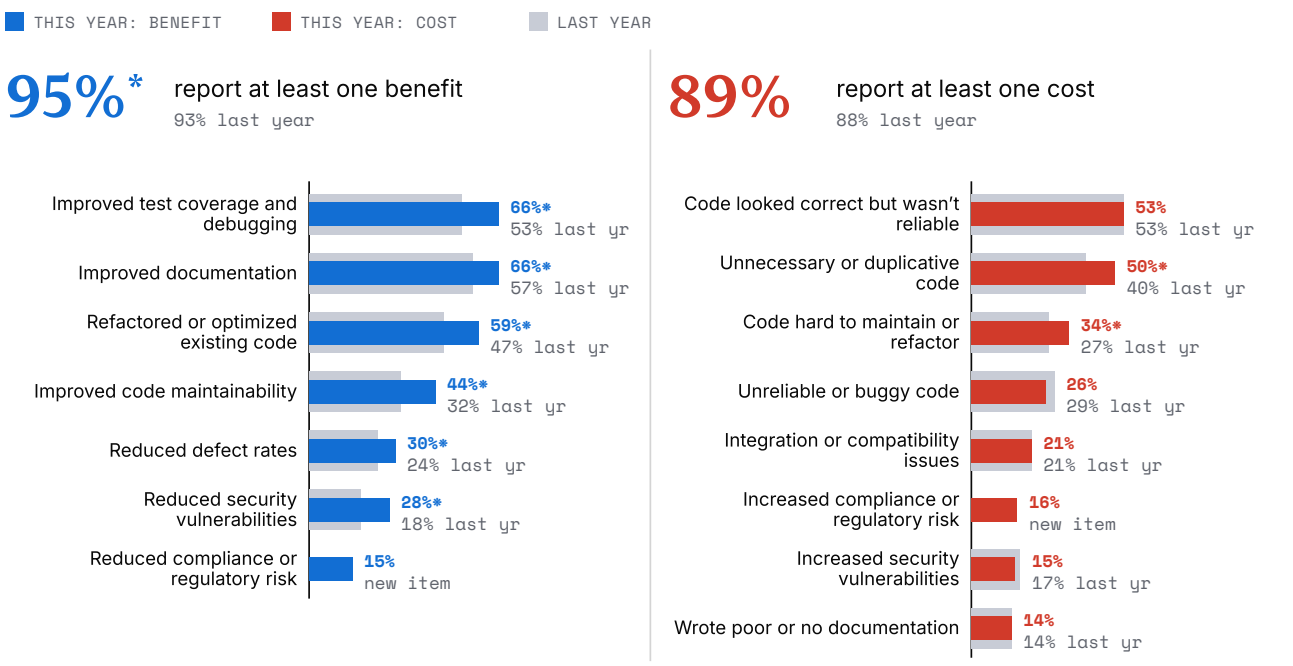


FIG. 36

We also asked about a variety of ways in which AI-generated code might impact organizations' technical debt. Overall, the results matched last year's survey. 95% of respondents report at least one positive impact of AI-generated code on their team's technical debt (93% last year). 89% report at least one negative impact (88% last year). In other words, nearly every team reports both positive and negative impacts on technical debt at the same time.

95% report at least one benefit of AI for technical debt, and 89% at least one cost

In which of the following ways, if any, has AI-generated code impacted your team / company's technical debt? | All benefits and costs asked about (compliance items new this year)



Last year: n=1,136 · This year: n=1,020 · *Significant change vs. last year

FIG. 37

While the negative impacts of AI on tech debt stayed fairly flat year over year, many of the positive impacts increased substantially. For example, improved test coverage and debugging rose from 53% to 66%, refactoring grew from 47% to 59%, and improved maintainability grew from 32% to 44%.

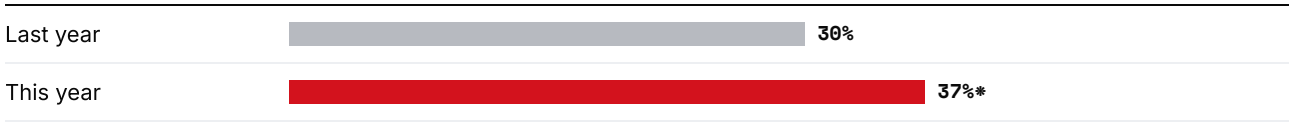
The most common negative effect, code that looked correct but was not reliable, held at 53%. Other negatives that did not change significantly include buggy code (29% to 26%) and integration problems (21%). The two that grew include:

- Unnecessary or duplicative code: 40% to 50%
- Code that is difficult to maintain or refactor: 27% to 34%

Concern about the long-term maintainability or readability of AI-generated code continued to rise, as 37% of respondents are very or extremely concerned, up from 30% last year.

Concern about long-term maintainability is rising

How concerned are you about each of the following when it comes to AI-generated code today? | Choice: Reduced long-term code maintainability or readability | Answer: Very or extremely concerned



Last year n=1,149 · This year n=1,026 · *Significant change vs. last year

FIG. 38

Tech debt shows up most where AI use is heaviest and review falls behind

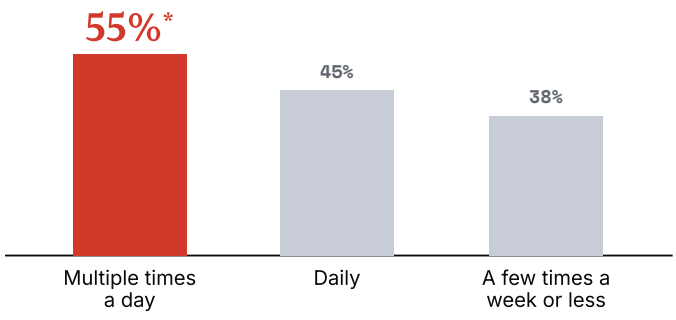
The more often teams use AI, the more often they see it pile up code. 55% of respondents whose teams use AI coding tools multiple times a day say it has produced unnecessary or duplicative code, compared with 45% of daily users and 38% of those who use it a few times a week or less.

The verification gap shows up here too. Respondents who say they generate AI code faster than they can verify it are more likely to report code that is difficult to maintain or refactor (38%) than those who do not (23%). That suggests the verification gap and maintainability problems tend to show up together.

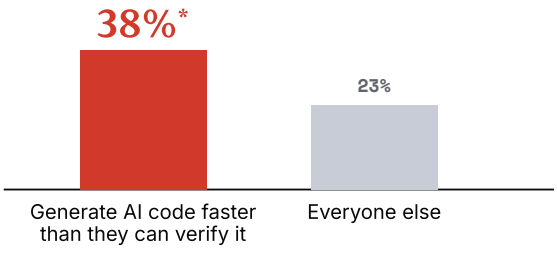
Tech debt builds up faster where generation outruns review

In which of the following ways, if any, has AI-generated code impacted your team / company's technical debt? · by AI usage frequency, and by agreement with "We generate AI code faster than we can verify it"

Report unnecessary or duplicative code, by how often the team uses AI



Report code that is difficult to maintain or refactor



Multiple times a day n=614 · Daily n=263 · Few times a week or less n=149 · Faster than verify n=780 · Everyone else n=238 (excl. N/A) ·
*Significant difference vs. daily users (left) and vs. everyone else (right)

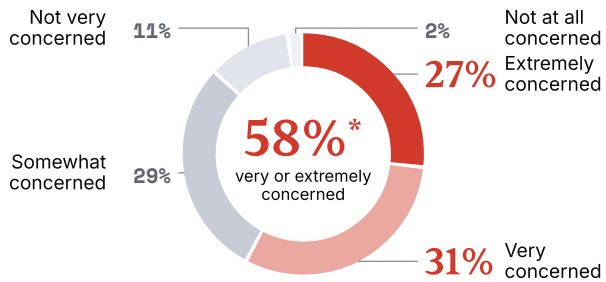
FIG. 39

The risk is not only to the codebase. 58% are very or extremely concerned that over-reliance on AI is eroding their team's understanding of the codebase (53% last year), and 53% worry about deskilling or erosion of their own coding abilities (48%).

The risk is not only to the codebase

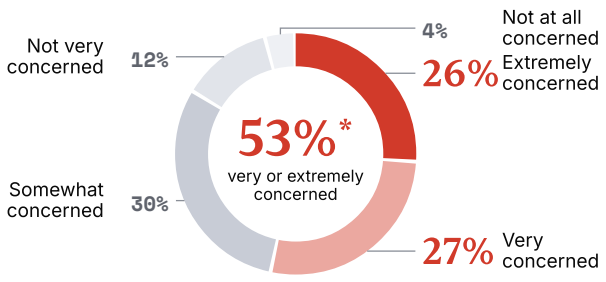
How concerned are you about each of the following when it comes to AI-generated code today?

Over-reliance erodes the team's understanding of the codebase



LAST YEAR, VERY / EXTREMELY CONCERNED: 53%

Deskilling or erosion of our own coding abilities



LAST YEAR, VERY / EXTREMELY CONCERNED: 48%

This year n=1,026 · Last year n=1,149 (full answer split not available for last year) · *Significant change vs. last year

FIG. 40

Over-reliance and deskilling are now the second- and fourth-ranked concerns overall. Respondents share concern that the knowledge needed to maintain the code is at risk alongside the code itself.

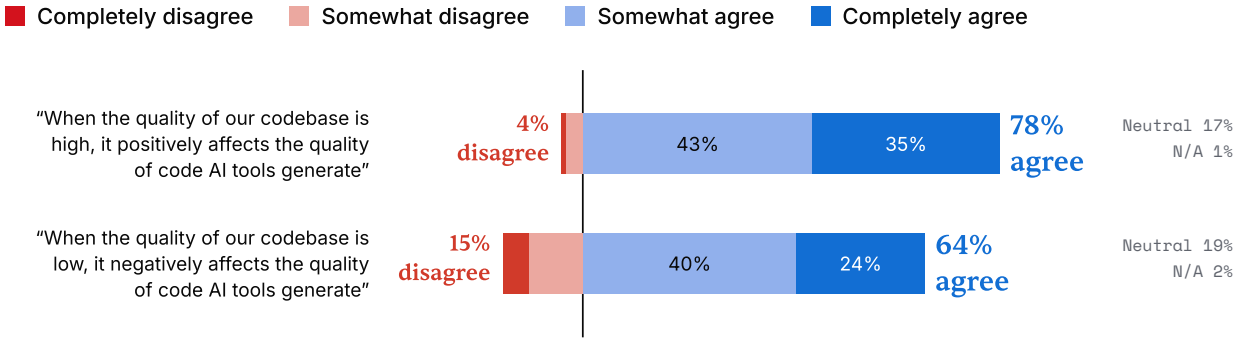
> **"It is way too easy to let [AI] produce bloated garbage that will solve the stated problem and let you close the ticket."**
>
> PRINCIPAL SOFTWARE ENGINEER · NORTH AMERICA · ENTERPRISE · HEALTHCARE

AI code mimics the code around it

Agents also learn from the codebase they are working in, and any added code can become context for the next change. Good coding practices or, conversely, a problem left in place today can shape many commits to come. 78% of developers and engineering managers agree that when the quality of their codebase is high, it positively affects the quality of code AI tools generate. But 64% agree that when quality is low, it makes AI output worse.

Codebase quality cuts both ways for AI-generated code

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools / agents?



n=1,026

FIG. 41

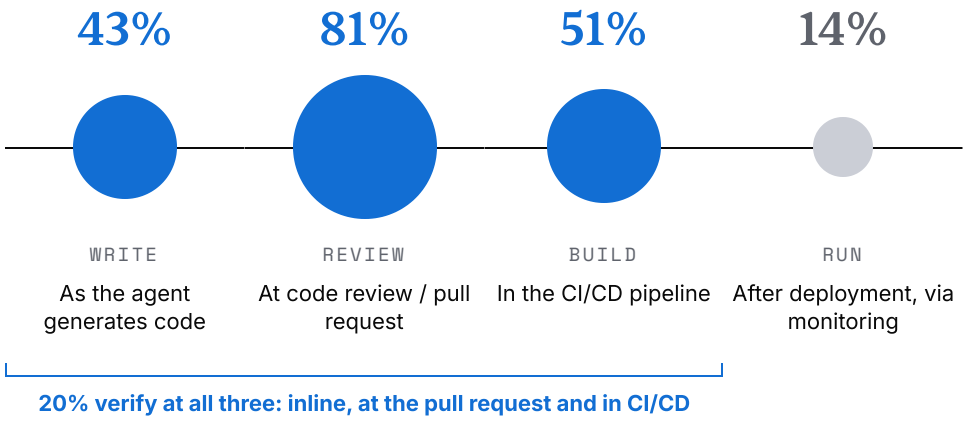
How AI code gets checked

Teams verify AI code mostly at review time and in pipelines, and less often as it is written. That means many issues surface late, when review is already the bottleneck.

Among respondents whose teams review any AI code, 81% say AI code gets analyzed or verified at code review or pull request time, 51% in the CI/CD pipeline, and 43% as the agent generates code, in real time. Only 20% do all three. 14% verify after deployment, through production monitoring, and 5% do not systematically verify AI-generated code.

Most AI code is checked at the pull request, fewer than half check it inline

At what point(s) in your workflow does AI-generated code get analyzed or verified? (select all that apply)



OUTSIDE THE LOOP

5%

do not systematically verify AI-generated code at all

Respondents whose teams review any AI code, n=1,021 · circle area proportional to share of respondents

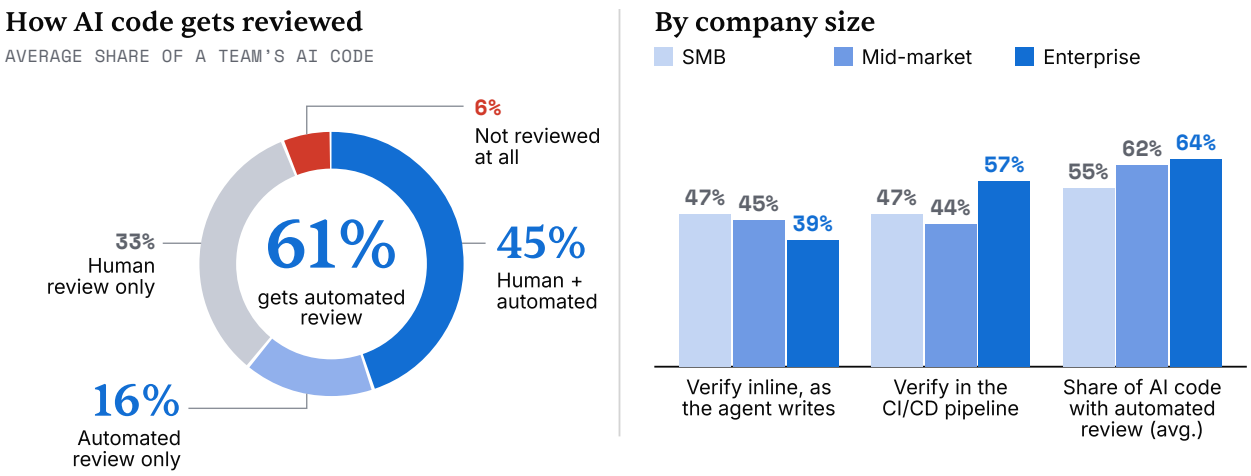
FIG. 42

Verification inside the agent's loop catches problems while the context is fresh and before they spread. Otherwise, by the time a change with an error reaches review, an agent may have built more code on top of it, thus magnifying the scope of the initial issue and the cost to fix it.

Respondents at small and midsize businesses (SMBs) are more likely to verify inline (47% vs. 39% at enterprises). Enterprises lean on CI (57% vs. 47%) and give more of their AI code automated review on average (64% vs. 55% at SMBs). Across all respondents, an average of 61% of a team's AI-generated code gets some automated review, alone or with a human.

Enterprises lean on CI and automation, SMBs on inline checks

What percentage of the code your team generates with AI is reviewed...? | At what point(s) in your workflow does AI-generated code get analyzed or verified? (select all that apply)



All respondents n=1,026 · SMB n=286 · Mid-market n=271 · Enterprise n=469

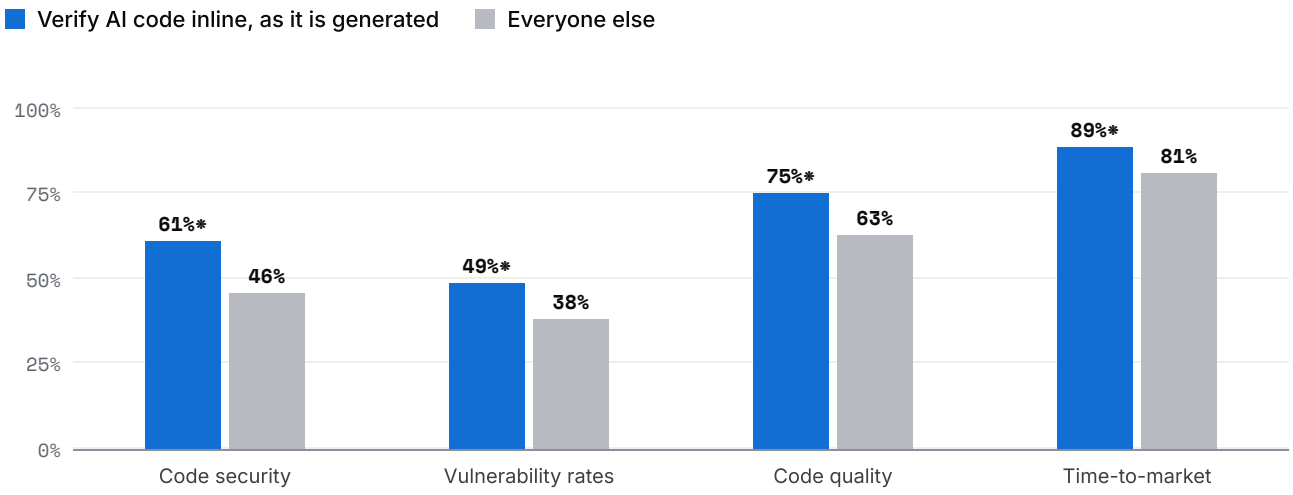
FIG. 43

Teams that check code as AI writes it report better security, quality, and speed

Teams that run automated review inline as their AI-generated code is written report stronger outcomes than teams that do not. 61% of them say AI has had a positive impact on code security, compared with 46% of teams without inline review. The gap holds for vulnerability rates (49% vs. 38%), code quality (75% vs. 63%), and time-to-market (89% vs. 81%).

Teams that verify AI code as it is written report stronger outcomes

What impact has AI-generated code had on your team / company for each of the following? | Answer: Somewhat or very positive impact



Verify inline n=438 · Everyone else n=583 · *Significant difference vs. everyone else

FIG. 44

> **"The biggest challenge is ensuring AI-generated code is consistently reviewed and validated, since it can introduce subtle bugs, security vulnerabilities, and inconsistent patterns at a much higher speed than humans can manually detect."**
>
> FRONT END TECH LEAD · LATIN AMERICA · SMB · TECHNOLOGY

THE TAKEAWAY

AI can reduce technical debt, and for most teams it does on balance. Teams that report the best results give it healthy code, relevant context, and automated review. Without these, faster generation can compound existing weaknesses.

That has a cost, and the next chapter looks at why the returns are not keeping up.

CHAPTER 6

Productivity is rising, but ROI is lagging behind

43% of respondents say AI coding has not yet delivered the return they expect

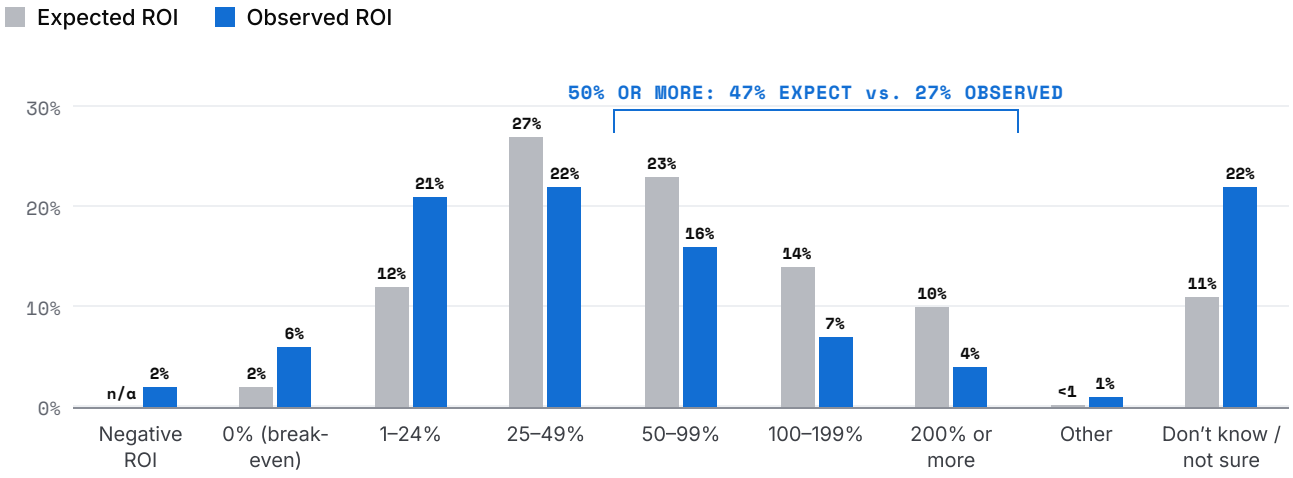
By many measures, AI development is increasing productivity. Half of respondents say AI makes them much more productive. More teams report faster delivery, higher quality, and better end-user experience than a year ago.

But some of the data points tell a different story. More than one in five respondents (22%) cannot say what return AI coding has delivered, and concern about AI costs rose more than any other concern we track.

47% of respondents expect an ROI of 50% or more from their AI coding tools, but only 27% report a return that high so far. Comparing each respondent's expected and observed return, 43% report a lower return than they expect, and 22% are unsure what return they have seen at all.

Teams expect more return from AI than they have seen so far

Approximately what return on investment does your team expect from its investment in AI coding tools? | ...has your team actually observed from its investment in AI coding tools?



50% or more: 47% expected vs. 27% observed

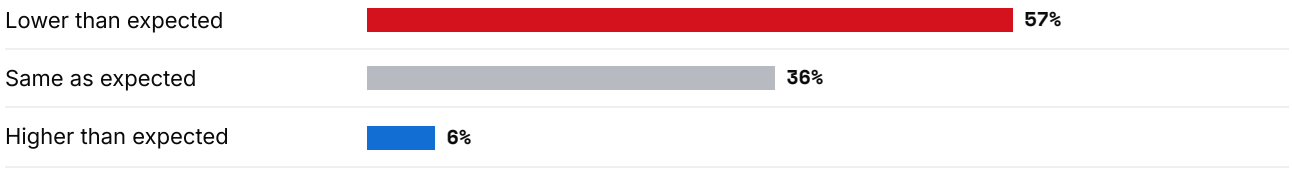
n=1,026

FIG. 45

Among the 75% of respondents who could estimate both their expected and observed return, 57% report a lower return so far than they expect, 36% the same, and 6% a higher one.

Most teams have seen a lower return from AI than they expect

Approximately what return on investment does your team expect from its investment in AI coding tools? | ...has your team actually observed? · each respondent's observed ROI band compared with their expected band



Respondents who gave a specific ROI band for both questions, n=770

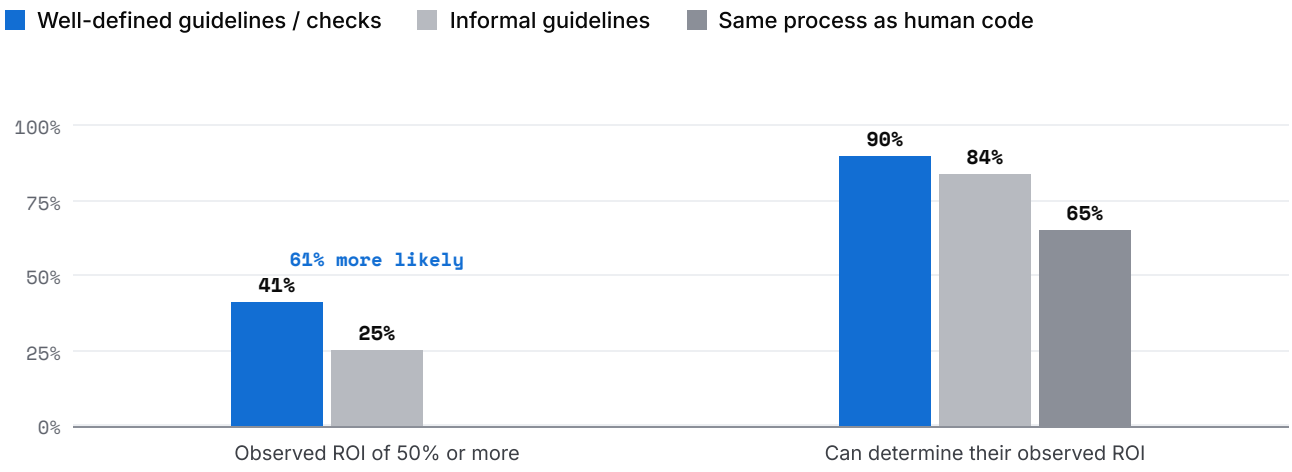
FIG. 46

> **"The industry needs better standards for measuring real productivity gains, verifying AI-generated code... rather than optimizing only for code volume."**
>
> DIRECTOR OF SOFTWARE ENGINEERING · EUROPE · ENTERPRISE · TECHNOLOGY

Teams that describe their guidelines as well-defined or automated are more likely to report an ROI of 50% or more (41% vs. 25% for teams with informal rules). They are also more likely to report an ROI estimate (90% vs. 84% with informal guidelines and 65% using the same review process as human code), suggesting greater visibility into returns.

Well-defined rules: 61% more likely to see an ROI of 50% or more

Approximately what return on investment has your team actually observed from its investment in AI coding tools? · by type of AI code guidelines



Well-defined n=239 · Informal n=330 · Same as human code n=248 · 61% from unrounded figures (40.6% vs. 25.2%)

FIG. 47

AI productivity is still rising

We asked respondents to estimate how much AI coding tools have changed their personal productivity, and how their gains have changed over the past year. Almost everyone who reported a change says they are more productive. Only 2% say AI has made them less productive.

The most common estimates are a gain of 21–40% (25%) and 41–60% (21%). For about a third (35%), the gain is double or more. 11% estimate 81–100% (about 2x), 9% three times, 8% four to five times, and 6% six times or more.

Also, the gains are not leveling off yet. 79% of respondents who report a productivity gain say it has kept growing over the past year, while only 17% say it has stayed the same or plateaued, and 2% say it has decreased.

For a third, AI at least doubled productivity, and gains are still growing

Please estimate the change in your personal productivity due to AI coding tools / agents. | In the past year, how have productivity gains from AI coding tools / agents changed?

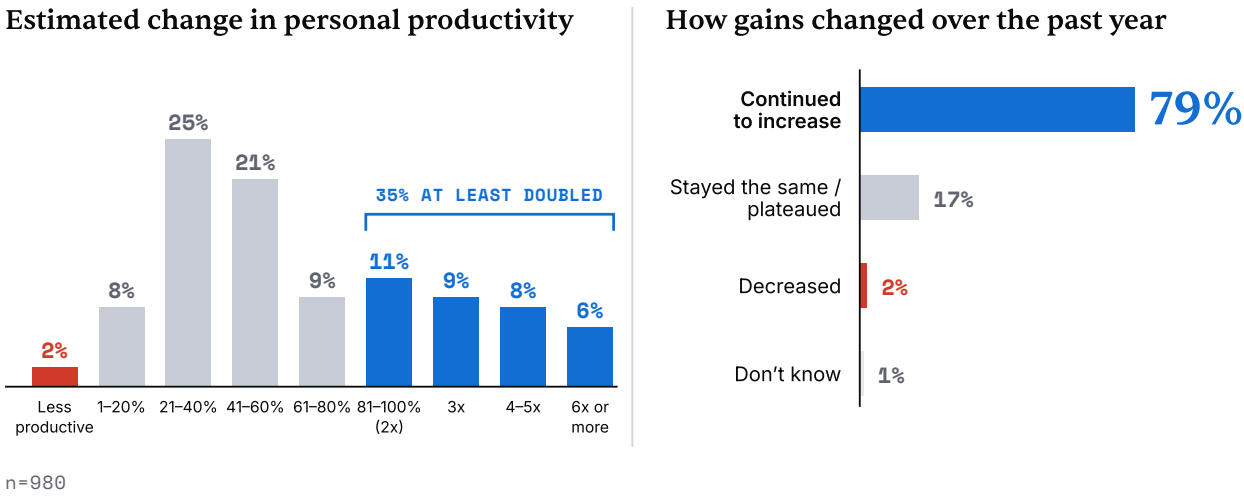


FIG. 48

36% of respondents say their team or company measures AI's impact on developer productivity and they have insight into it, up from 22%. The share of respondents not measuring at all fell from 50% to 35%.

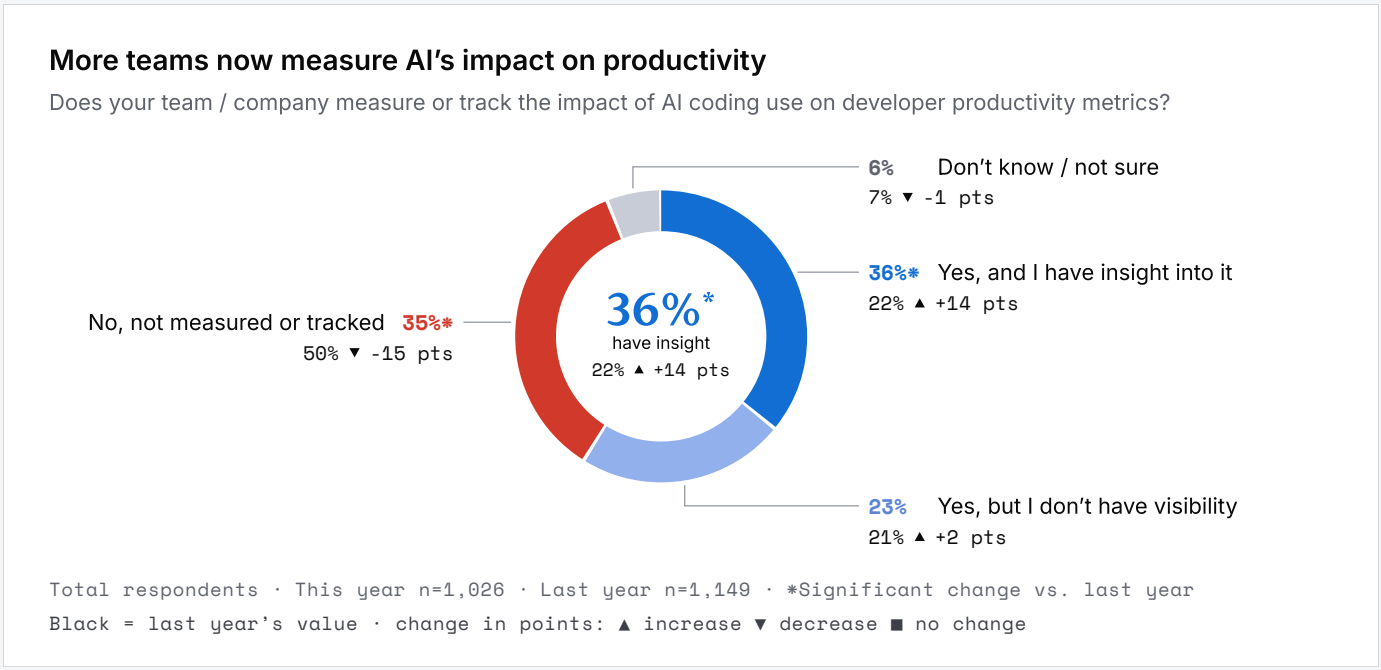


FIG. 49

Sentiment and reported impact improved on most measures

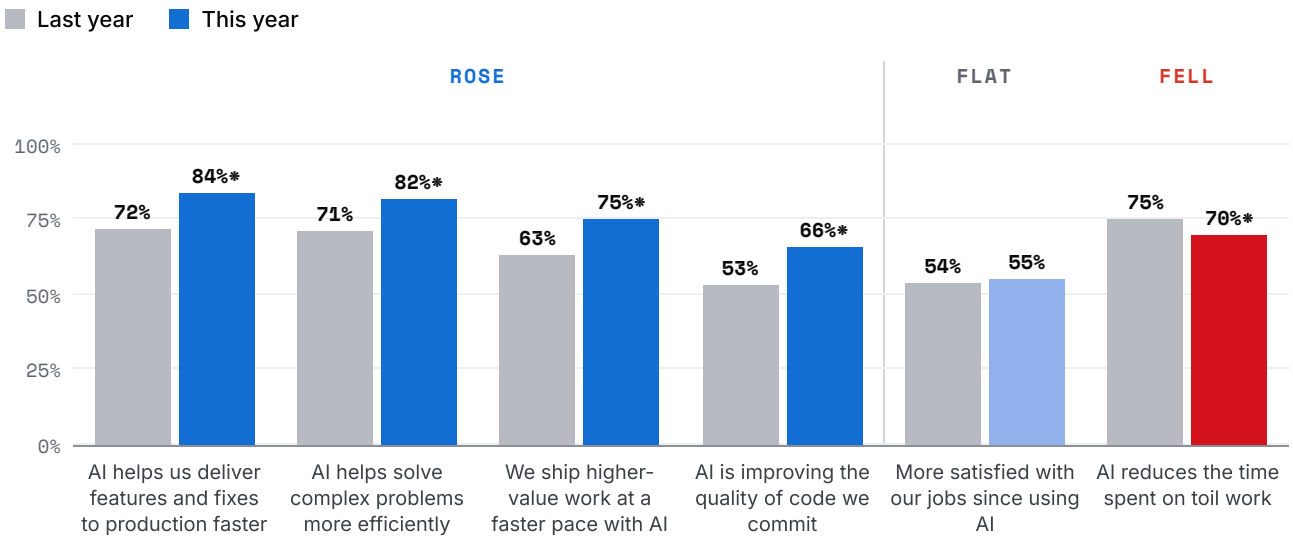
Agreement rose on most positive statements about AI in the past year:

- AI helps us deliver features and fixes to production faster: 72% to 84%
- AI helps solve complex problems more efficiently: 71% to 82%
- We ship higher-value work at a faster pace with the help of AI: 63% to 75%
- AI is improving the quality of code we commit: 53% to 66%

Job satisfaction did not move much (54% to 55%), and agreement that AI reduces toil fell (75% to 70%). Respondents have also reported increased business impact from AI-generated code, and most impacts (10 out of 12) improved significantly.

Agreement rose on most positive statements about AI

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools / agents? % agree somewhat or completely

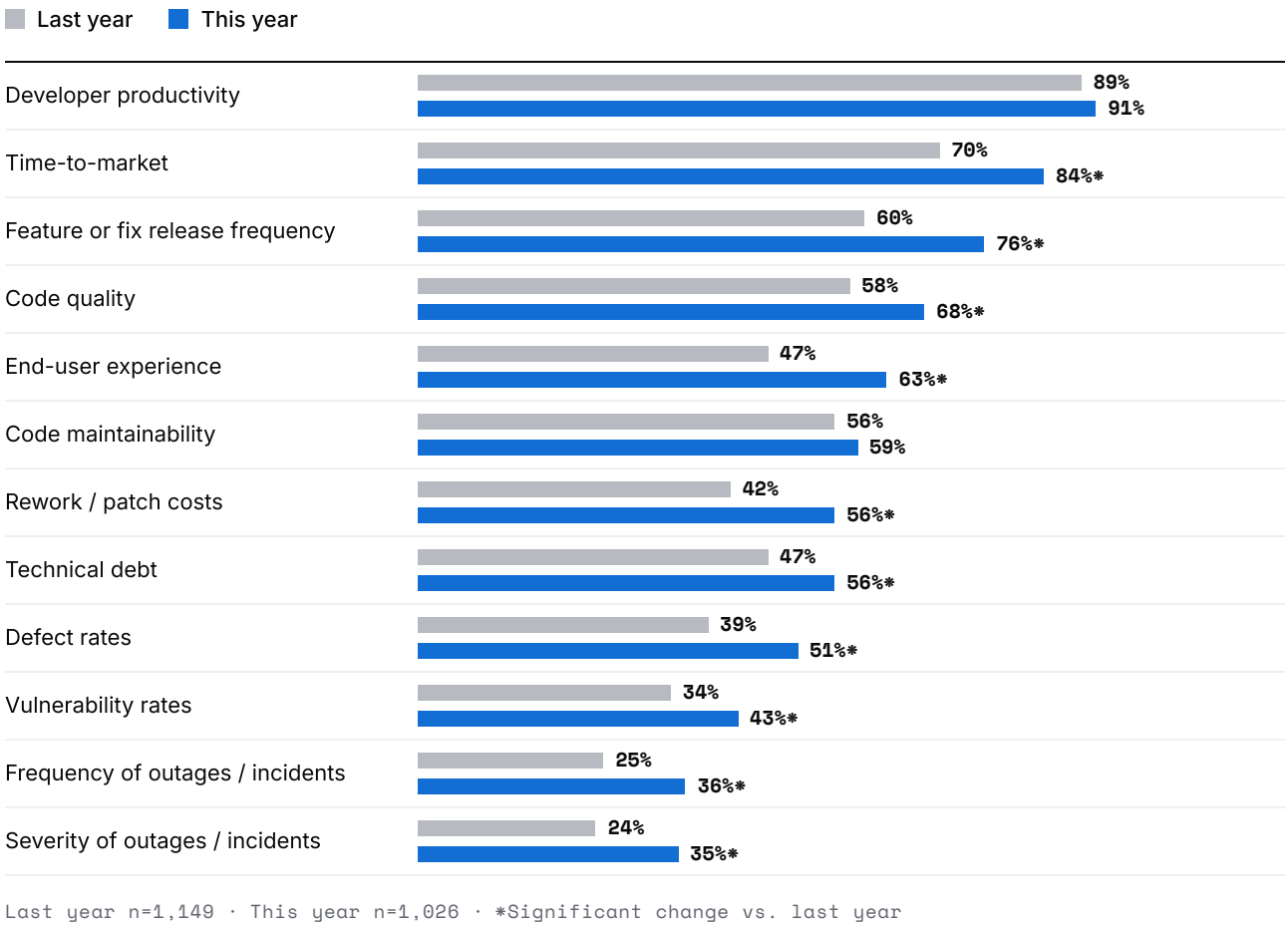


Individual contributors and managers combined · This year n=1,026 · Last year n=1,149 · *Significant change vs. last year

FIG. 50

Reported impact of AI rose on all 12 outcomes, significantly on 10

What impact has AI-generated code had on your team / company for each of the following? | Answer: Somewhat or very positive impact · sorted by this year



Last year n=1,149 · This year n=1,026 · *Significant change vs. last year

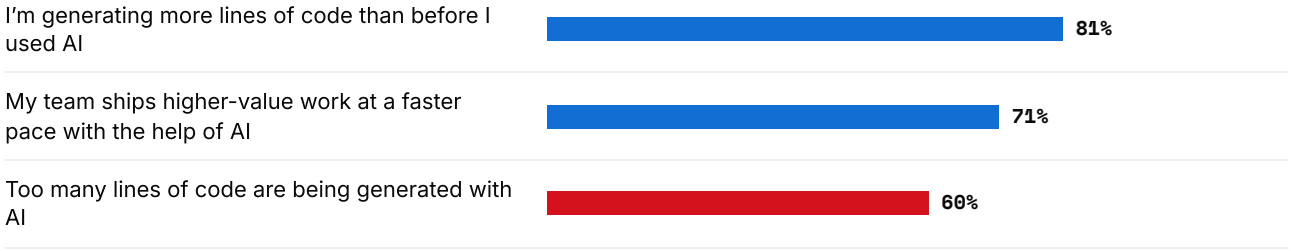
FIG. 51

The largest gains are in delivery and quality. Time-to-market rose from 70% last year to 84% now, release frequency improved from 60% last year to 76% now, end-user experience from 47% to 63%, and defect rates from 39% to 51%.

Output and value are not rising together. Among respondents whose main job is writing code, 81% agree they generate more lines of code than before AI, and 71% agree their team ships higher-value work faster. But 60% also say too many lines of code are being generated.

More code is easy to see, more value is harder to prove

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools / agents? % agree somewhat or completely · respondents whose main job is writing code



Respondents who primarily write code and do not manage, n=270

FIG. 52

Toil has not shrunk

We also asked how much of their work week goes to toil, the friction that saps productivity or adds frustration. Respondents estimate about 27% of their work is toil work (24% last year), while 54% say toil takes more than a quarter of their week (up from 38% last year).

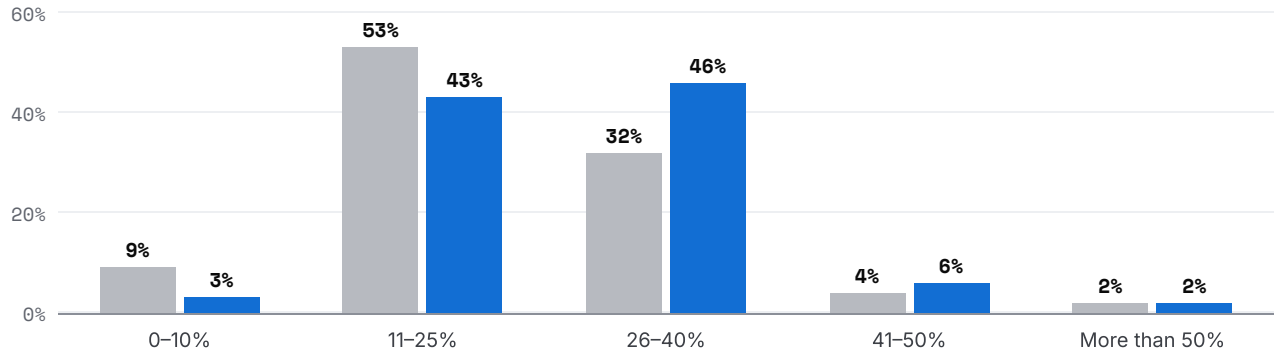
Toil now takes more than a quarter of the week for most respondents

Approximately what percentage of your typical work week do you estimate is spent dealing with workflow friction ("toil work") that hinders productivity or increases frustration in your current role?

Last year This year

27%* average share of the week spent on toil, up from 24% last year

54%* spend more than a quarter of the week on toil, up from 38% last year



More than 25%: 38% last year vs. 54% this year

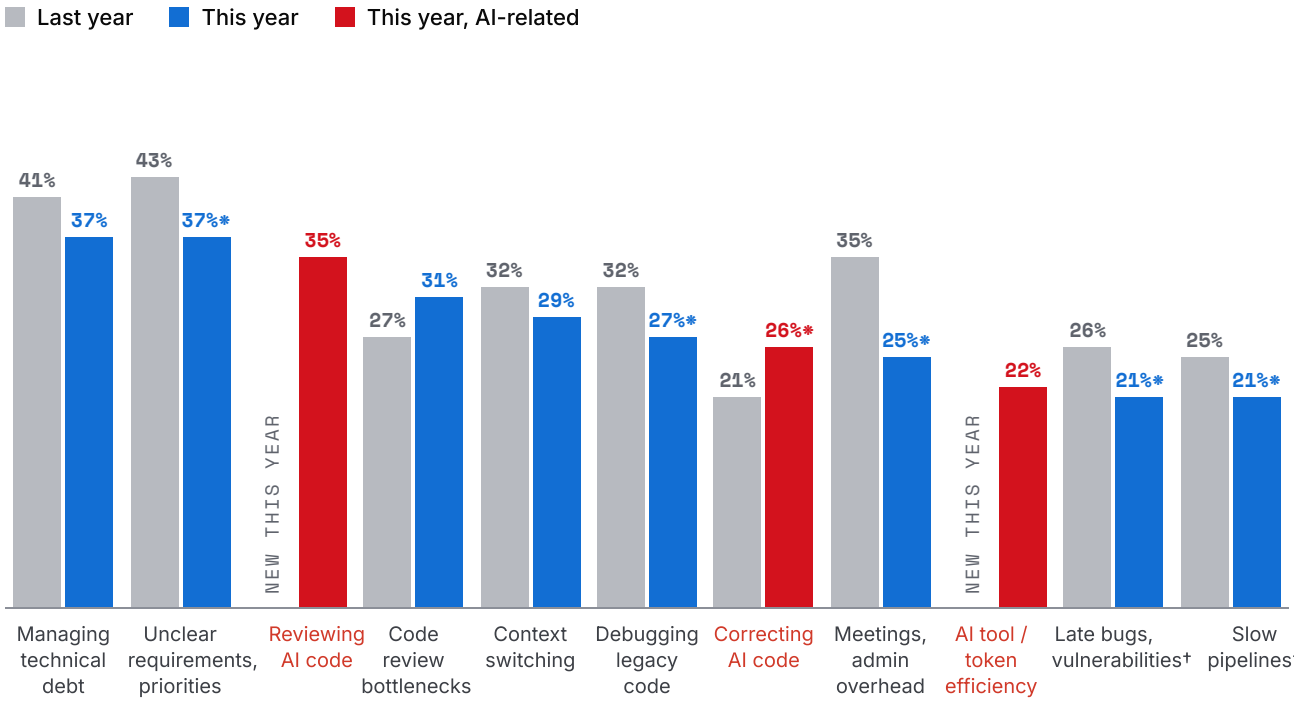
Respondents who selected any source of toil · This year n=1,026 · Last year n=1,149 · *Significant change vs. last year

FIG. 53

As we observed last year, some of the toil work is shifting toward managing AI's output. Reviewing code created by AI coding tools is one of the top three sources of toil (35%), alongside unclear requirements (37%) and managing technical debt (37%). Correcting or rewriting AI-generated code was reported by 26% of respondents as a source of toil.

Reviewing AI's code is one of the top three sources of toil

Thinking about your current workflows, what are your biggest sources of friction ("toil work") that hinders productivity or increases frustration in your current role? · top 10 this year · select all that apply



Total respondents · This year n=1,026 · Last year n=1,149 · †Tied for 10th (21% each) · *Significant change vs. last year

FIG. 54

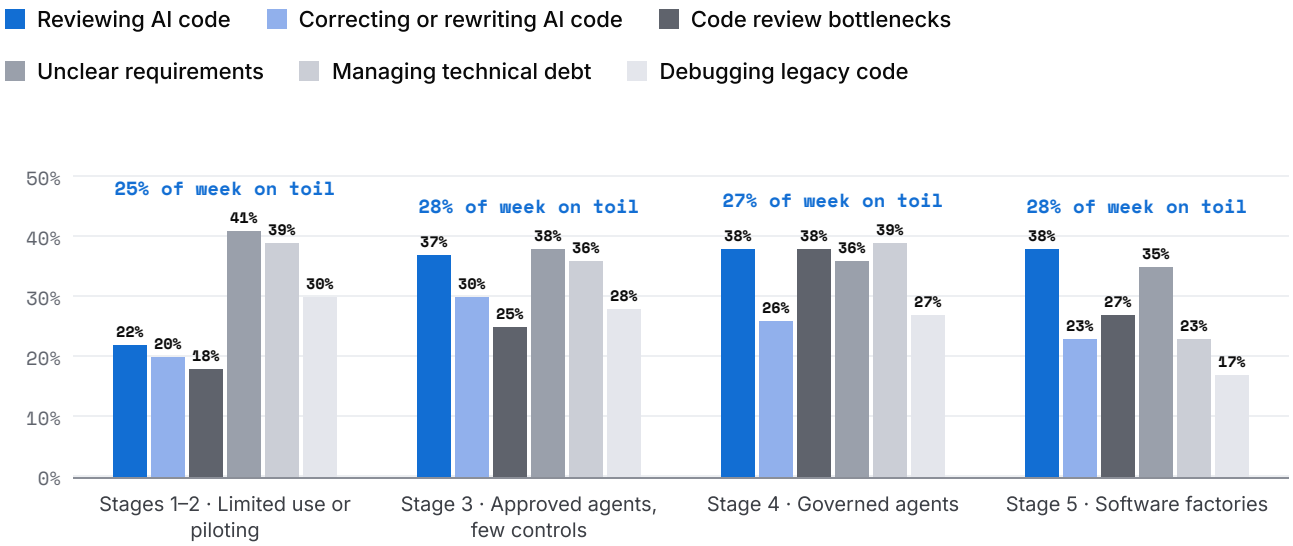
The share of the week lost to toil barely moves as teams adopt AI more fully. Respondents at every stage of AI adoption, from limited use and piloting (Stages 1-2) to building autonomous software factories (Stage 5), put it at between 25% and 28% of their week. What changes is where that time goes.

At the earliest stages, toil still looks familiar. Unclear requirements (41%) and managing technical debt (39%) lead, and only about one in five name reviewing or correcting AI-generated code. But once developers are using agents, reviewing AI code jumps to 37% and correcting or rewriting it peaks at 30%. This fits teams that have capable agents but few standards to steer them.

With governed agents, correcting AI code eases to 26% as a source of toil, but the pressure shifts onto the review process itself, and code review bottlenecks rise from 25% at Stage 3 (company-approved agents with few controls) to 38%. Among the limited group already running software factories today, older friction such as debugging legacy code (17%) and managing technical debt (23%) falls well below other stages, while reviewing AI code stays at 38%.

As AI adoption deepens, toil moves toward checking what agents produce

What are your biggest sources of toil? | Approximately what percentage of your typical work week is spent dealing with toil?
by current level of AI adoption · % of respondents



Total respondents by current AI adoption stage · Stages 1–2 n=152 · Stage 3 n=290 · Stage 4 n=535 · Stage 5 n=48 (small base)

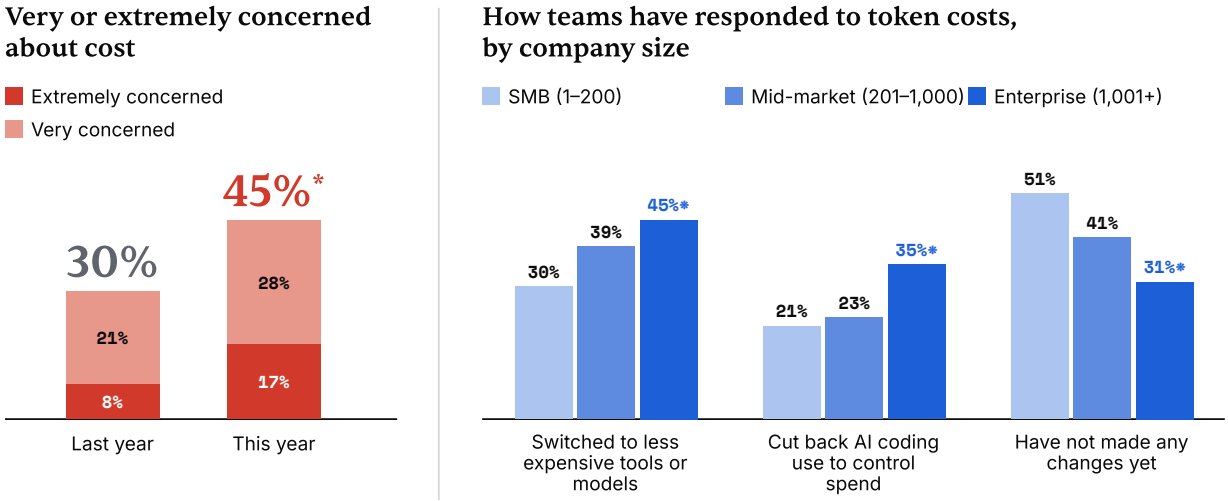
FIG. 55

Cost became a constraint

Concern about cost rose more than any other concern we track. 45% are very or extremely concerned about the cost of advanced AI coding tools and agents, up from 30%.

Cost concern is rising, and larger companies are already acting on it

How concerned are you about the cost associated with using advanced AI coding tools / agents? | Which of the following best describes how your team has responded to token costs?



Total respondents · Cost concern: this year n=1,026, last year n=1,149 · Token response (select all): SMB n=286, mid n=271, ent. n=469 · *Significant change vs. last year (left) or vs. SMB (right)

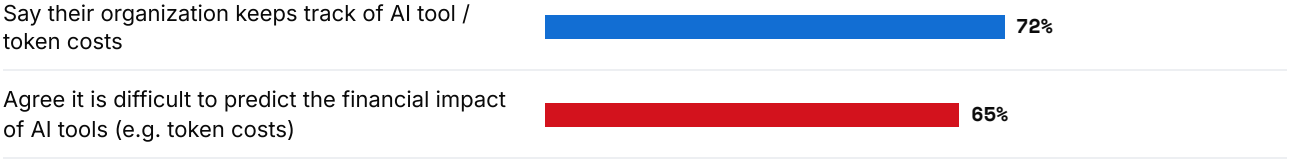
FIG. 56

How teams respond to token costs depends heavily on company size. Among enterprises, 45% have switched to less expensive tools or models and 35% have cut back AI coding to control token spend, compared with 30% and 21% of small and midsize businesses (SMBs). Mid-market companies sit in between, with 39% switching to less expensive tools or models and 23% cutting back. Smaller companies are the most likely to have held off: 51% of SMBs have not made any changes yet, compared with 41% of mid-market companies and 31% of enterprises.

But tracking and predicting costs effectively is a pain point. 72% say their organization tracks AI tool and token costs, yet 65% of developers and engineering managers agree it is difficult to predict the financial impact of AI tools such as token costs.

Most teams track AI costs, yet most still find them hard to predict

Does your organization keep track of AI tool costs, including token costs? | To what extent do you agree: It is difficult to predict the financial impact (e.g. token costs) associated with using AI tools · % agree somewhat or completely



Total respondents, n=1,026

FIG. 57

> **“The biggest challenge is the ever-changing cost and uncertainty around how much prompts will cost.”**
>
> SOFTWARE ENGINEERING MANAGER · NORTH AMERICA · ENTERPRISE · TECHNOLOGY

THE TAKEAWAY

Developers feel faster, and teams report stronger business impact. But sentiment, output, and ROI are three different measures, and most organizations still lack the visibility to connect them and realize benefits all the way through. As costs become harder to ignore, the teams that can demonstrate returns will find it easier to keep their budgets.

The next chapter looks at how the tools themselves, and the money behind them, are shifting.

CHAPTER 7

AI investments are shifting and diversifying

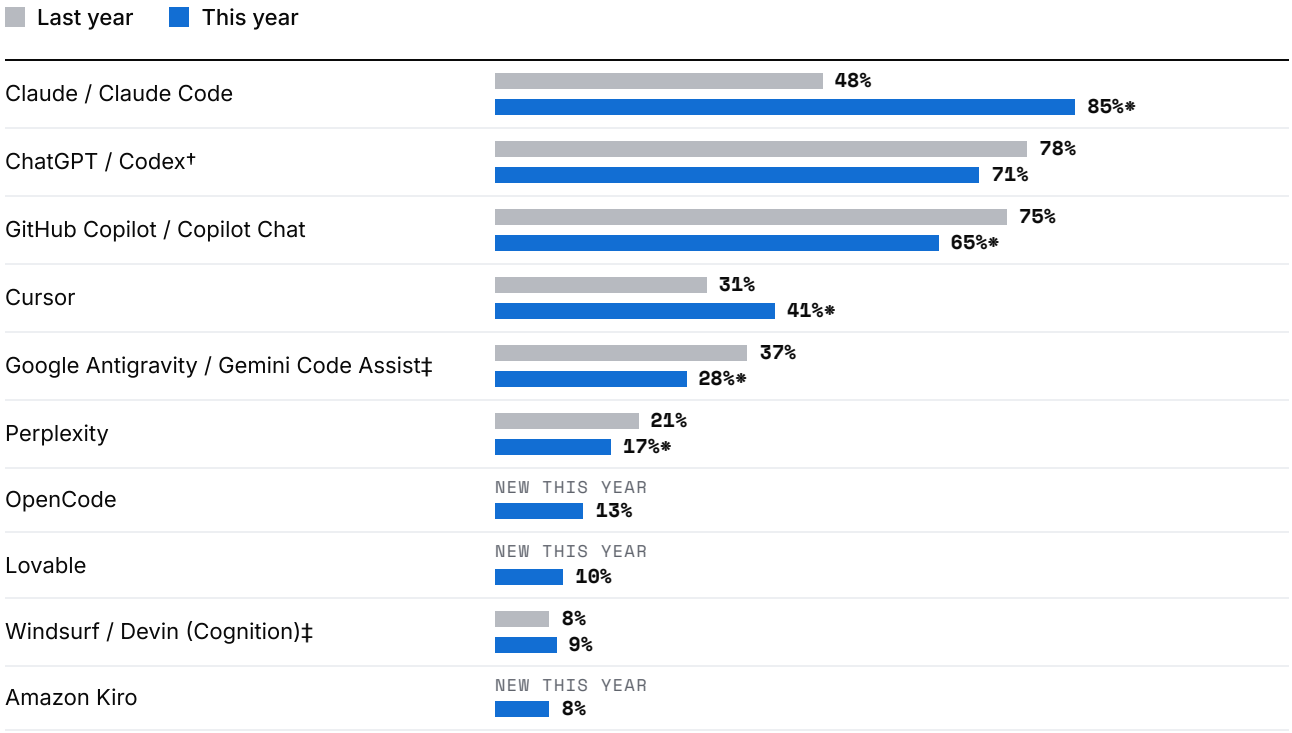
AI coding solutions have seen big changes in the last year

The leading tools have changed, personal accounts have given way to work-sanctioned ones, and teams have started planning model portfolios instead of single-vendor bets. The common thread is choice: more tools, more models, and more ways to host them. That is good for teams, but it complicates the job of standardizing what all of them produce in a consistent way.

85% of respondents say they or their team used Claude/Claude Code for software development in the past year, up from 48% and the biggest jump of any tool. GitHub Copilot fell from 75% to 65%, and Cursor rose from 31% to 41%.

Claude is now the most used AI coding tool, overtaking ChatGPT and Copilot

In the past year, which of the following AI coding tools have you or your team / company used for software development tasks? Top 10 this year · select all that apply



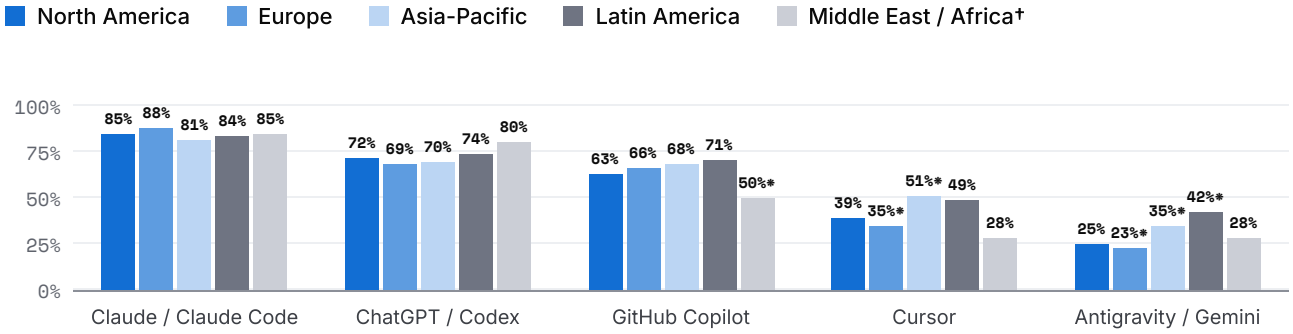
†This year's figure combines ChatGPT and OpenAI Codex, which were listed separately last year (ChatGPT alone: 74%, Codex alone: 21%, combined 78%).
‡Answer wording changed: last year listed Gemini Code Assist and Windsurf.
Total respondents · This year n=1,026 · Last year n=1,149 · *Significant change vs. last year

FIG. 58

Claude/Claude Code, ChatGPT/Codex, and GitHub Copilot lead adoption in every region, in nearly the same order, but the biggest regional differences appear among the smaller tools. Cursor is much more popular in Asia-Pacific (51%) and Latin America (49%) than in Europe (35%) or the Middle East/Africa (28%), and Antigravity/Gemini follows a similar pattern, reaching 42% in Latin America compared with 23–25% in Europe and North America.

The same five tools lead in every region

In the past year, which of the following AI coding tools have you or your team / company used for software development tasks? Top 5 tools · % of respondents in each region



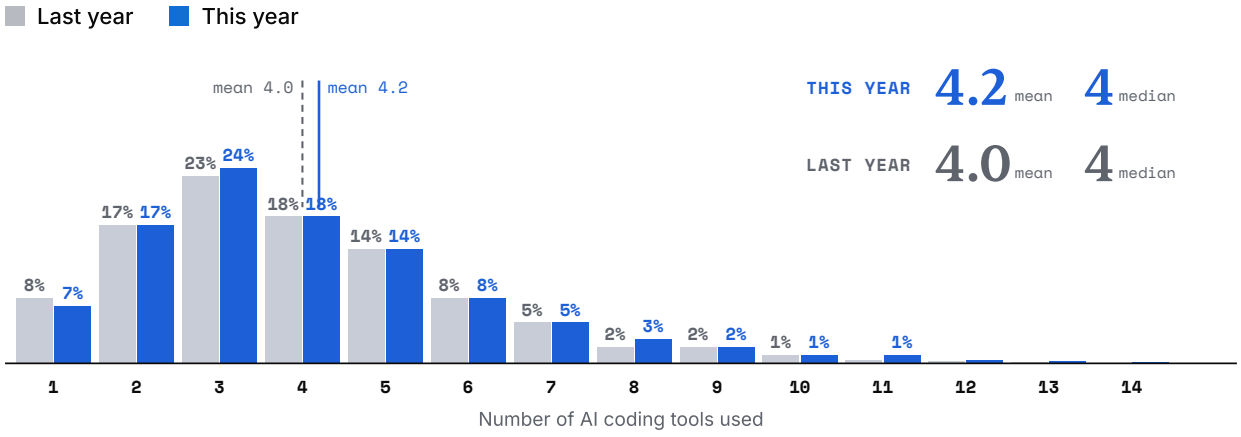
North America n=468 · Europe n=229 · Asia-Pacific n=220 · Latin America n=69 · Middle East / Africa† n=40 (small base) · *Significant difference vs. all other regions

FIG. 59

Respondents use about 4.2 AI coding tools on average, similar to last year's 4.0, and the median held at four tools.

Respondents use about as many AI coding tools as last year

In the past year, which of the following AI coding tools have you or your team / company used for software development tasks? Number of tools selected · % of respondents



Total respondents · This year n=1,026 · Last year n=1,149 · Values under 0.5% not labeled · 0.6% of respondents this year used 15 or more tools (not shown)

FIG. 60

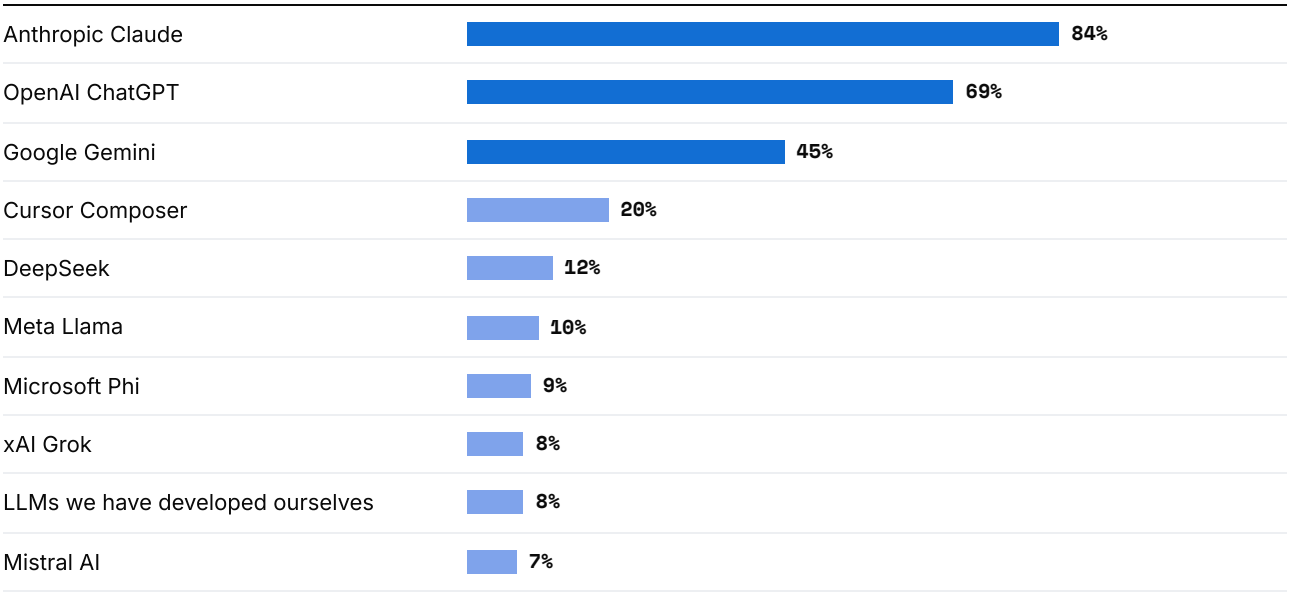
> **“I believe the single biggest challenge is stopping the amount of extra AI tools you introduce...The complexity of the CI/CD pipeline can become higher, introducing conflicts and disagreement between models and tools.”**
>
> IT & COMPLIANCE MANAGER, PRODUCT OWNER · ASIA-PACIFIC · MID-MARKET · CONSULTING

Models used largely mirror tools

84% say their team uses Anthropic Claude models for software development, 69% OpenAI ChatGPT models, and 45% Google Gemini. Smaller shares use DeepSeek (12%), Meta Llama (10%), Microsoft Phi (9%), xAI Grok (8%), their own LLMs (8%), and Mistral AI (7%).

Model choice correlates with tool choice

Which of the following models are you or your team / company currently using for software development tasks? Top 10 · select all that apply



Total respondents, n=1,026 · New question this year

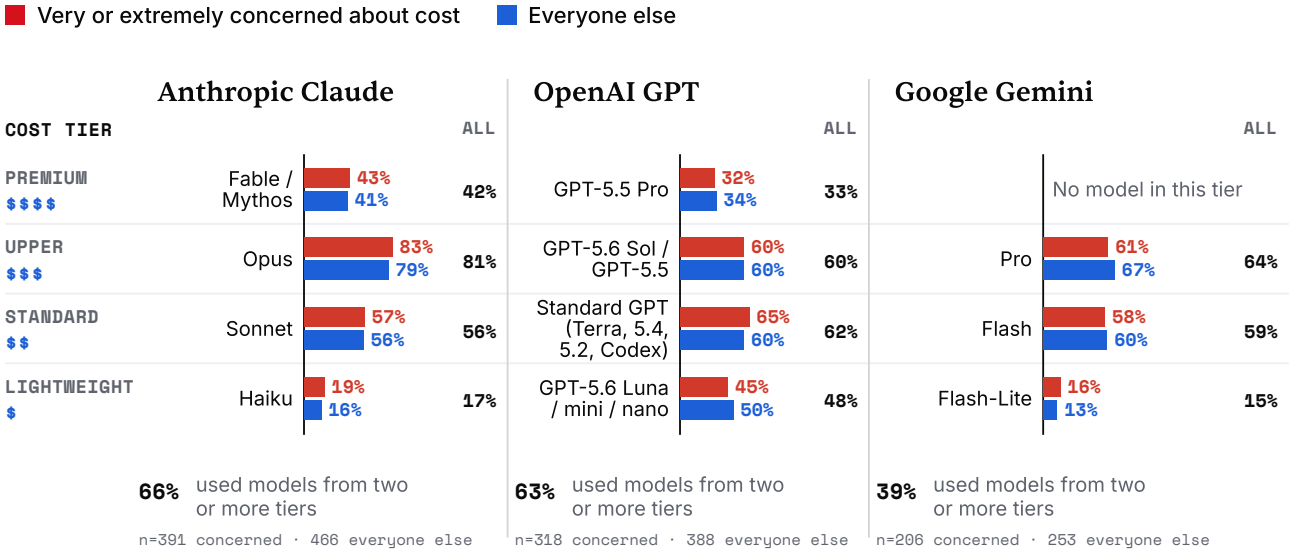
FIG. 61

The data also shows that Anthropic users lean toward the upper tier model rather than the premium or lower ones: 81% used Opus, compared with 42% for Fable or Mythos, and 56% for Sonnet. OpenAI and Gemini users split their work more evenly between the upper (60% for OpenAI, 64% for Gemini Pro) and standard (62% for OpenAI, 59% for Gemini Flash) tiers. OpenAI usage is also relatively heavy on lightweight models: 48% used GPT-5.6 Luna, mini, or nano, which is much higher than others (17% for Anthropic Haiku, 15% for Gemini Flash-Lite).

Concern about cost does not appear to change which models respondents choose. In every tier, respondents who are very or extremely concerned about cost are within six points of everyone else. Most Anthropic and OpenAI users also mix tiers: 66% and 63% used models from two or more, compared with 39% of Gemini users.

Cost-concerned teams choose model tiers much like everyone else

Which of the following LLM models have you used in the past 3 months? · grouped by comparable price tier
% of each provider's users who used at least one model in the tier



Users of each provider's models: Anthropic n=857 · OpenAI n=706 · Google n=459 · Select all that apply · Tier grouping is Sonar's

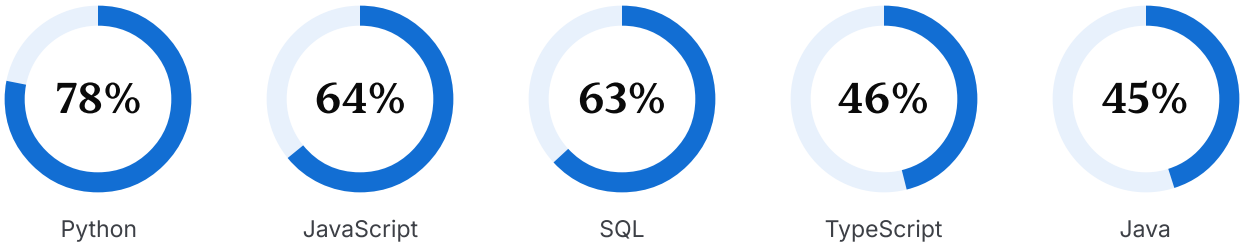
FIG. 62

The language most used with AI coding tools is Python

Python is the most popular language for AI coding: 78% of respondents have used it extensively in the past year. JavaScript (64%) and SQL (63%) follow, with TypeScript (46%) and Java (45%) rounding out the top five.

Python is the language developers use most with AI

Which of the following programming languages have you used extensively when producing code with AI tools in the past year? Top 5 languages · select all that apply



Total respondents, n=1,026

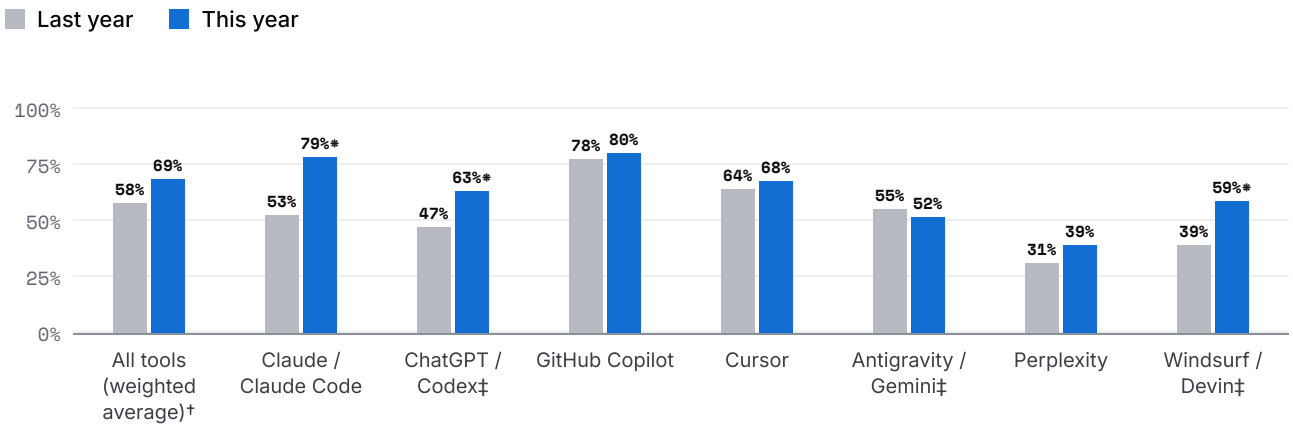
FIG. 63

Shadow AI is shrinking, which gives businesses more control

Last year, a surprising (and concerning) percentage of developers reported using AI tools on personal accounts. That is changing as more organizations embrace AI coding. The share of respondents accessing AI tools through a work-sanctioned account or license rose for most of the leading tools, and now averages 69%, up from 58% last year (weighted by how many people use each tool). Among the leading tools, the biggest jumps came for Claude, from 53% to 79%, ChatGPT/Codex, from 47% to 63%, and Windsurf (listed as Windsurf/Devin this year), from 39% to 59%.

Shadow AI is giving way to work-sanctioned accounts

In the past year, how have you personally used each of the following AI coding tools for software development tasks? % "through a work-sanctioned / owned account or license" · among those whose team uses each tool



†Weighted by how many people use each tool, across tools asked in both years. ‡Last year asked as "ChatGPT", "Gemini Code Assist" and "Windsurf"; Codex and Devin were added to the answer options this year.

Respondents whose team uses each tool; bases vary by tool · This year n=1,026 · Last year n=1,147 total · *Significant change vs. last year

FIG. 64

This is good news for governance and security. Unlike a tool on a personal account, work-sanctioned tools can be configured, monitored, and connected to the team's standards with less risk of leaking business data.

Teams plan to mix closed- and open-weight models

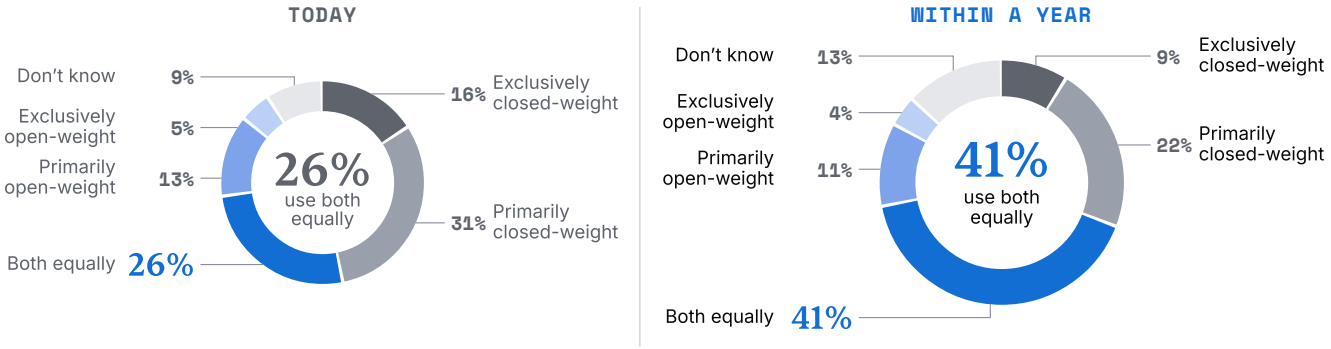
We also asked which types of models teams use for software development now, and which they plan to use in a year. 41% of respondents expect their organization to use open- and closed-weight models equally within a year, compared with 26% who use them equally today.

CLOSED- VS. OPEN-WEIGHT MODELS

Closed-weight models have trained weights that are not publicly available to download or modify, while open-weight models have trained weights that are publicly available to download and locally modify.

More organizations expect to balance open- and closed-weight models

Which of the following best describes the types of AI models your organization currently uses / plans to move towards within a year for software development tasks?



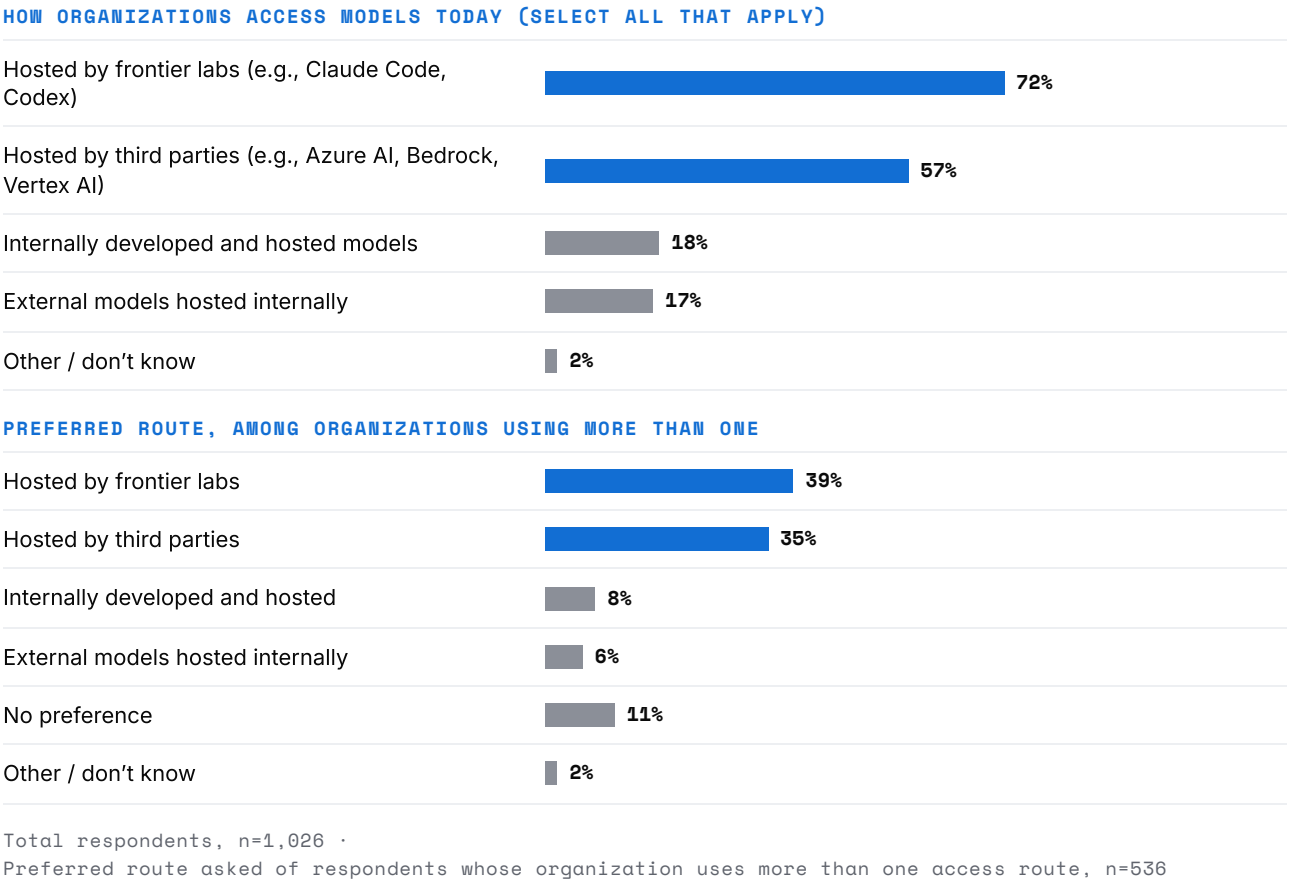
Total respondents, n=1,026

FIG. 65

Teams are hosting the models they use in diverse ways. 72% access models hosted by frontier labs (e.g. Anthropic, OpenAI), and 57% through third-party hosts such as Microsoft Azure AI, AWS Bedrock, or Google Vertex AI. Among respondents whose organizations use more than one hosting route, 39% prefer frontier labs and 35% third-party hosts.

How teams reach AI models is diversifying

How does your organization currently access AI models for software development? | What is your organization's preferred way to access AI models for software development?



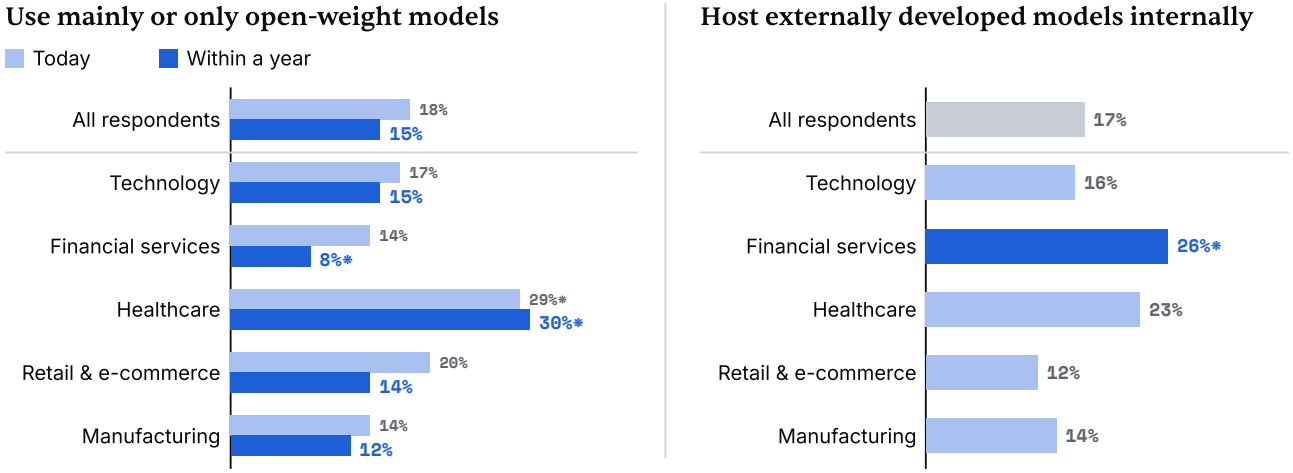
Total respondents, n=1,026 ·
Preferred route asked of respondents whose organization uses more than one access route, n=536

FIG. 66

Regulated industries are taking different routes to maintain control over their models. Healthcare respondents are the most likely to run mainly or only open-weight models today (29% vs. 18% on average), and twice as likely to plan a move in that direction within a year (30% vs. 15%). Financial services is the least likely to plan that move (8%), but the most likely to host externally developed models internally (26% vs. 17%), keeping control by running models in-house.

Regulated industries take different routes to control over their models

Which best describes the types of AI models your organization currently uses / plans to move towards within a year? | How does your organization currently access AI models for software development?



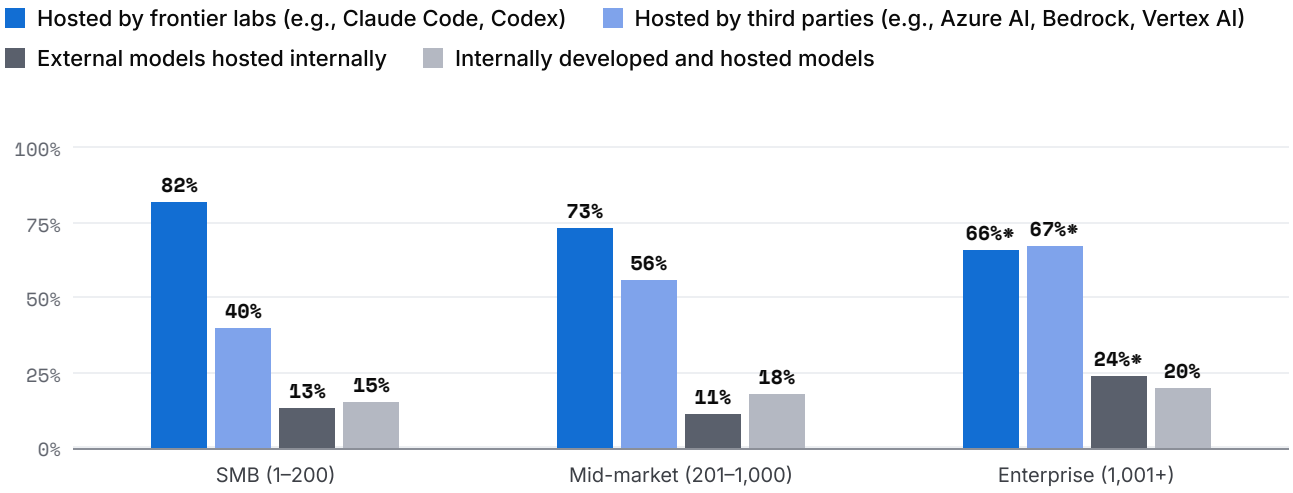
All n=1,026 · Technology n=601 · Financial services n=106 · Healthcare n=69 · Retail n=49 · Manufacturing n=43 · Bases under 70: directional · *Significant difference vs. all other industries

FIG. 67

Hosting choices also vary by company size as 67% of enterprise respondents use third-party hosting, vs. 40% at small and midsize businesses (SMBs), while 82% of SMB respondents go directly to frontier labs, vs. 66% at enterprises. Enterprises are also more likely to host externally developed models internally (24% vs. 13%).

The larger the company, the more it relies on third-party hosting

How does your organization currently access AI models for software development?



Total respondents by company size · SMB n=286 · Mid-market n=271 · Enterprise n=469 · Other / don't know (under 2%) not shown · *Significant difference vs. SMB

FIG. 68

A portfolio of models, hosts, and tools is a sensible hedge on cost and capability. It also means the same codebase could be changed by several models with different habits. Verification that is independent of any one model is what keeps quality consistent when the model underneath changes.

> **“The biggest concern that I have is the model of AI we are utilizing currently and then being forced to upgrade models because the older model is no longer supported. We are moving so quickly that our models could be compromised and we may not even know it.”**

ARCHITECTURE LEADER · NORTH AMERICA · ENTERPRISE · TECHNOLOGY

THE TAKEAWAY

The strategy behind AI coding stacks is less concentrated and more deliberate than a year ago. That flexibility raises a practical question: how can we verify and guide code in a way that works regardless of which model wrote it?

The next chapter looks at teams that use SonarQube for that job and how their outcomes compare.

CHAPTER 8

SonarQube: Ship agentic code with confidence

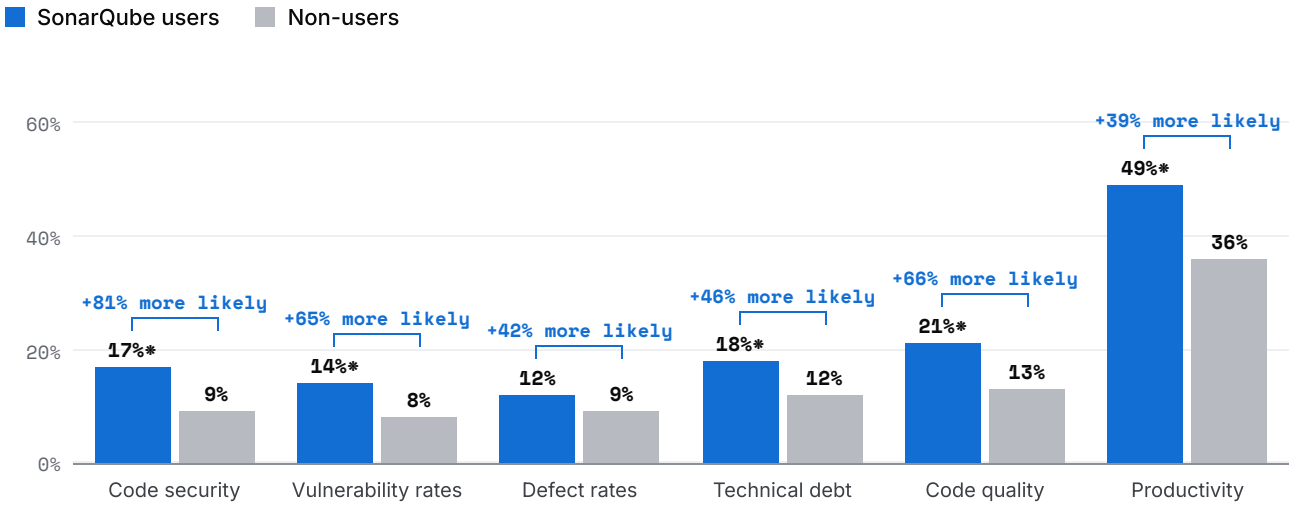
SonarQube users see better outcomes from AI coding

Agents now write most of the code, but engineers need the confidence to ship it. Alongside the general AI questions, we asked respondents whether they use SonarQube. The data shows SonarQube users are more likely to have that confidence, and more likely to say that AI has had a very positive impact on security, quality, and productivity. They are also more likely to have the practices this report keeps returning to: well-defined guidelines, systematic context for agents, verification in CI, and collection of measurements and evidence.

Our data shows that SonarQube users are 81% more likely to say AI-generated code has had a very positive impact on code security compared to non-users. Similar gaps emerge on vulnerability rates (65% more likely to report very positive impacts), defect rates (42% more likely), technical debt (46% more likely), and code quality (66% more likely). They are also 39% more likely to report very positive impacts on their productivity.

SonarQube users are far more likely to report that AI strongly improved their code

What impact has AI-generated code had on your team / company for each of the following? | Answer: Very positive impact · "more likely" relative difference shown above each pair



SonarQube users: n=319 · Non-users: n=707 · *Significant difference vs. non-users

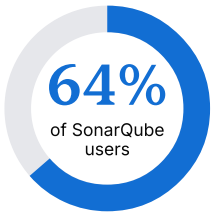
FIG. 69

SonarQube users who connect to their AI coding tools (the way Sonar Vortex is designed to do) report the strongest outcomes of any group we compared. 64% of SonarQube users connect it to their AI coding tools. Compared with non-users, they are 130% more likely to say AI-generated code has had a very positive impact on code security, 116% more likely to say this for vulnerability rates, 112% for code quality, 83% for technical debt, 70% for defect rates, and 59% for developer productivity.

SonarQube users who connect to their AI tools see stronger results

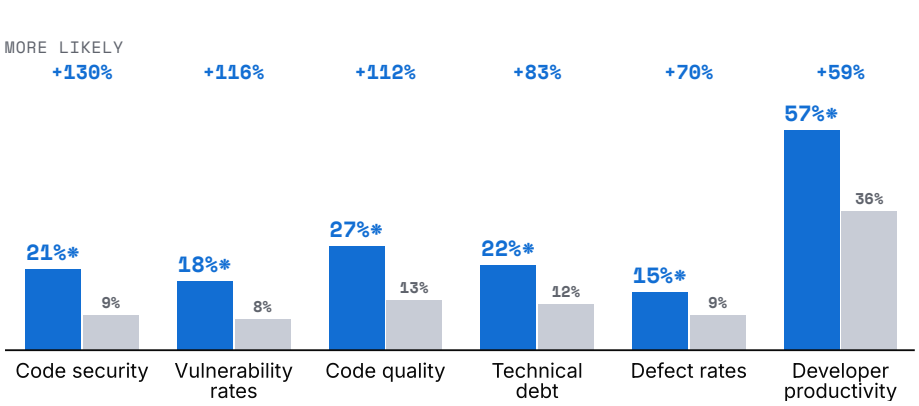
Do you use any of the following to connect SonarQube to your AI coding tools? | What impact has AI-generated code had on your team / company for each of the following? · % very positive impact

Connect SonarQube to AI coding tools



via MCP, CLI or Sonar Vortex

Very positive impact of AI-generated code



SonarQube users n=319 · Connected SonarQube users n=205 · Non-users n=707 · % diffs from unrounded figures · *Significant vs. non-users

FIG. 70

SONAR VORTEX

Sonar Vortex is an AI governance layer that holds any coding agent to your standards, as it writes code. It delivers your architecture, coding standards, security policy, and approved libraries to the agent before it writes a line, so the change fits your design on the first attempt and completes coding projects using fewer tokens. Every agentic change is checked in real time, inside the agent's loop, against the same SonarQube quality profile and security standards that already gate your merges.

> "[M]aintaining high standards requires...tight integration with quality/security analysis tools like SonarQube."

SENIOR INDIVIDUAL CONTRIBUTOR · NORTH AMERICA · MID-MARKET · TECHNOLOGY · SONARQUBE USER

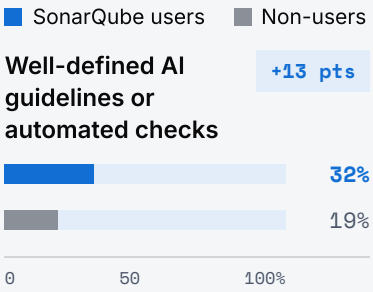
Guidance and verification

What do SonarQube users do differently? They are more likely to have the guidance and verification practices that link to stronger reported outcomes:

01 / SET THE RULES

Intent

Teams define what good AI-generated code looks like before it is written.



HOW SONAR HELPS

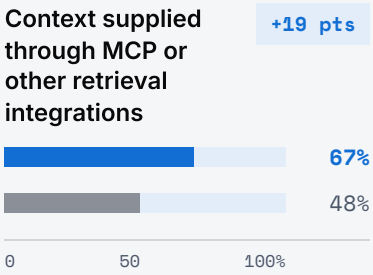
Quality profiles | Quality gates | Architecture guiderails

Set the structural and release requirements, so the relevant standards are enforced on every change.

02 / GUIDE THE AGENT

Guiderails

Agents get your architecture, standards and context while they work, not after.



HOW SONAR HELPS

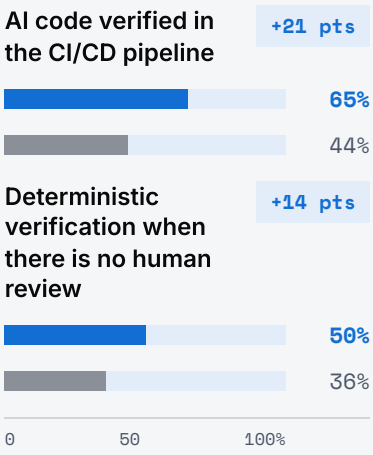
Sonar Vortex | MCP | CLI | Plugins

Give coding agents your architecture, standards, security policy, and approved libraries before they write code, guiding them as they work. You can also connect Sonar via MCP, CLI, plugins, and other available integration options.

03 / CHECK EVERY CHANGE

Verification

Every change is checked, whether or not a person reviews it.



HOW SONAR HELPS

Multilayered analysis | Sonar Vortex | Hunter Agent | Remediation Agent

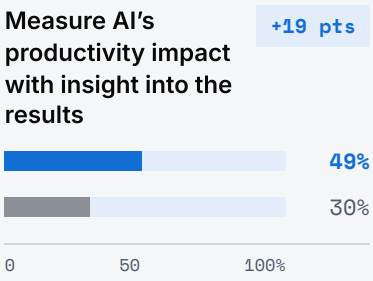
Multilayered analysis can check every change, in the agentic loop with Sonar Vortex, at the pull request, and in CI/CD, against the release controls set at the quality gate.

SonarQube Hunter Agent scans the codebase for logic-level issues, and the SonarQube Remediation Agent supplies verified fixes for issues in the backlog.

04 / SHOW THE PROOF

Evidence

Teams can show how controls were set up and applied across the codebase.



HOW SONAR HELPS

Security & compliance reports | Quality gate history | Portfolio views | Audit logs | Remediation history

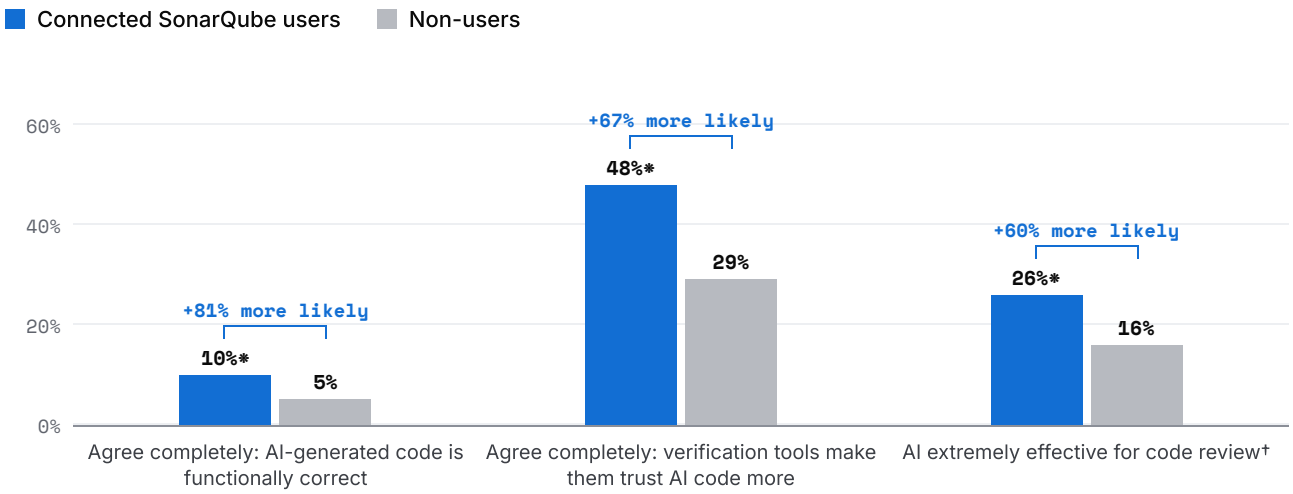
Teams get a record of how controls were set up and applied across the entire codebase.

SonarQube helps build confidence in AI output

SonarQube users who connect to their AI tools are more likely to have confidence in AI-generated code. They are 81% more likely than non-users to agree completely that it is functionally correct, 67% more likely to agree completely that strong code verification tools make them trust AI code more, and 60% more likely to rate AI extremely effective for code review.

SonarQube users who connect to their AI tools trust AI code more

To what extent do you agree with each of the following statements as it relates to the use of AI coding tools / agents? How effective are AI coding tools / agents for... code review?



Connected SonarQube users n=205 · Non-users n=707 ·
†Among teams using AI for code review: n=158 vs. 512 · *Significant difference vs. non-users

FIG. 71

> **“AI tools accelerate speed, but enterprise security demands deterministic gates like SonarQube to prevent technical debt. Developers must master architecture and governance to direct AI outputs safely”**
>
> ENGINEERING MANAGER · ASIA-PACIFIC · MID-MARKET · TECHNOLOGY · SONARQUBE USER

WHAT THIS MEANS

AI does not fix a broken process; it amplifies what is already there. In a well-architected system with strong guiderails and verification, agents amplify efficiency. In a low-trust, unpredictable one, they amplify the chaos and produce more “looks correct but isn’t” code, faster. Sonar guides and verifies every change independently, inside the agent’s loop, at the pull request, and in CI, so agent-written code meets the team’s quality and security standards before it reaches production.

SonarQube users, who invest more in guidance, verification, and measurement, report stronger productivity, quality, and security outcomes from AI tools. And the strongest results are reported by the users who connect SonarQube directly to their AI tools.

CHAPTER 9

What leaders should do next

Power your agentic software factories to ship with confidence

This report began with a simple contrast: code is abundant, but confidence is scarce. AI now writes or assists with most code, while volume keeps climbing. And a third of respondents expect to run software factories within a year, which will accelerate code creation even more. This means that code generation continues to outpace verification, manual checks are slipping, and returns on investment are hard to prove. The teams with the strongest outcomes build confidence in their AI-generated code from the start.

We believe organizations can maximize the confidence in the code created by their agents with a simple formula:

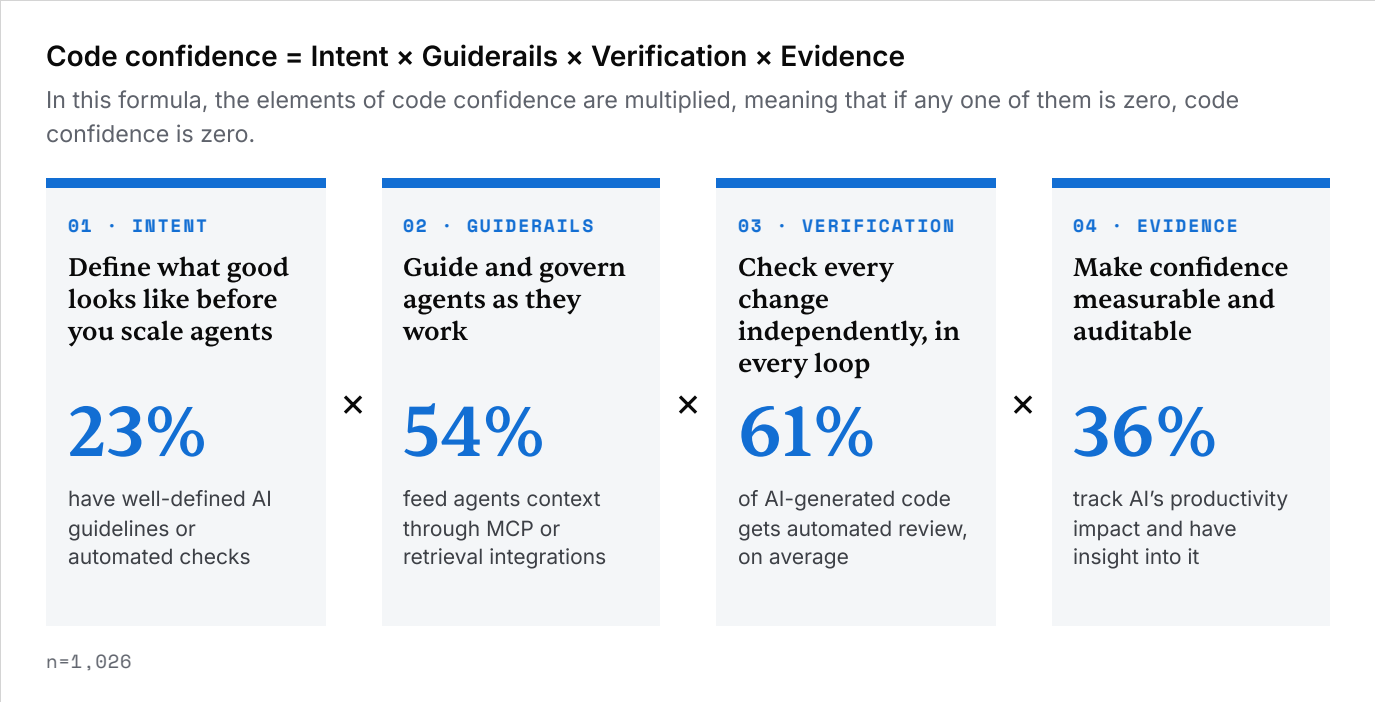


FIG. 72

It is important to note that in this formula, each of the elements of code confidence is multiplied, meaning, if any one of them is zero, then code confidence is zero. Providing excellent context with no independent verification still ships unchecked code. Rigorous review with no written standards has nothing consistent to check against.

Here is how software engineering teams can increase code confidence by investing in each element of the formula:

1. Intent: Define what good looks like before you scale agents

An agent can only meet a standard that has been written down, and at agent volume, reviewers cannot enforce unwritten rules consistently.

- Write your informal rules down as standards and architecture, specific enough to check against.
- Give agents specs to implement, and use the same specs as the reference that reviews and automated checks measure against.
- Set security requirements before agents start writing code, so verification has a clear bar to hold changes to.
- Hold non-engineers who build software with AI to the same standards.

Do you have well-defined, AI-specific guidelines enforced by automated checks?

23% of survey respondents do



n=1,026

FIG. 73

2. Guiderails: Guide and govern agents as they work

Teams with agents approved by their companies but no guiderails report the most risk, and teams with governed agents report better outcomes. For example, they report stronger code security and are significantly more likely to report ROI of 50% or more.

- Give agents codebase context and quality guiderails through a shared, systematic process, so the quality of their input does not depend on who writes the prompt.
- Choose harnesses deliberately, whether that means one standard for everyone or different harnesses for different purposes by design.
- Put controls on tokens, permissions, and tool access, and route each task to the model best suited to it.
- Keep the codebase healthy, because this code is the input AI learns from and builds on.

Do you feed agents codebase context systematically, and choose agent harnesses deliberately?

54% supply context through MCP servers or other retrieval integrations

48% standardize on one harness or use different harnesses by design

n=1,026

FIG. 74

3. Verification: Check every change independently, in each loop

Human code review can no longer scale to agentic volume. Automated verification can check that code meets standards without a person reviewing every change. Running it independently means the check does not come from the model that wrote the code, so it does not share that model's blind spots.

- Verify inline as the agent writes, at the pull request, and in CI.
- Enforce a standard quality gate on all AI-generated code, protecting your coding standards through multilayered agentic and deterministic verification.
- Never let a model review its own work.
- Keep review rigor steady as agents take on more autonomy.

What share of your AI-generated code gets automated review?

61% of AI-generated code gets automated review, on average

22% gets no separate human review

Averages of reported percentages · n=1,026

FIG. 75

4. Evidence: Make confidence measurable and auditable

Teams cannot defend what they do not document, so every change needs a record of what was checked and what the checks found.

- Keep a record of every quality gate and agent action, so leaders, auditors, and regulators can see what was checked and why it passed.
- Measure AI's impact on productivity and ROI before you scale it further.
- Track review load as a core metric, alongside output.
- Make token spend forecastable, so cost does not become a surprise that slows adoption down.



FIG. 76

Final thoughts

The best factories in history were engineered for quality and consistency, and agentic software factories will be no different. Because of agents, producing code is no longer a constraint. But when an agent writes the code that runs the business, the people – not the agents – are accountable. And the safeguards our people have relied on for decades were not designed for work arriving at this speed and volume.

Confidence has to be engineered into the factory. That starts with giving agents clear intent, so they know the outcome they are working toward and the limits they must respect. It means guiderails that keep agents on course while they work. It demands verification that checks every change independently before it ships. And it requires evidence that clearly shows what was checked, found, and fixed. Getting all four of these right builds the confidence we need to accelerate AI coding adoption safely.

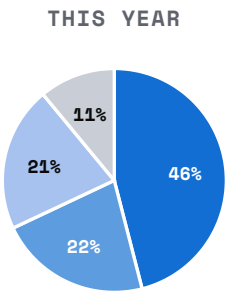
Code is abundant. Confidence is scarce. It does not have to stay that way.

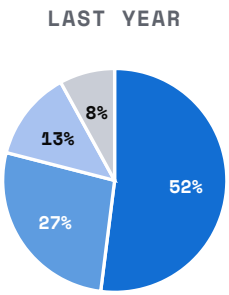
Appendix: About our survey demographics

We surveyed 1,026 developers and engineering managers who use AI coding tools at work. 91% are senior individual contributors, managers, or directors and above, and 83% write code. All figures are shares of all respondents (2027 n=1,026; 2026 n=1,149) unless otherwise noted.

Region

Share of respondents by region





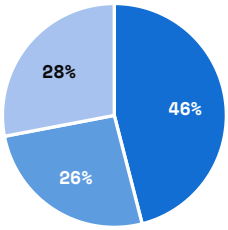
	THIS YR	LAST YR
North America (US, CA, MX)	46%	52%
Europe	22%	27%
Asia-Pacific	21%	13%
Other (Latin America, Middle East / Africa)	11%	8%

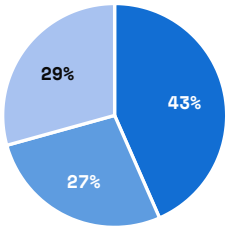
Total respondents · This year n=1,026 · Last year n=1,149

FIG. 77

Company size

Approximately how many employees work at the organization you are currently with?





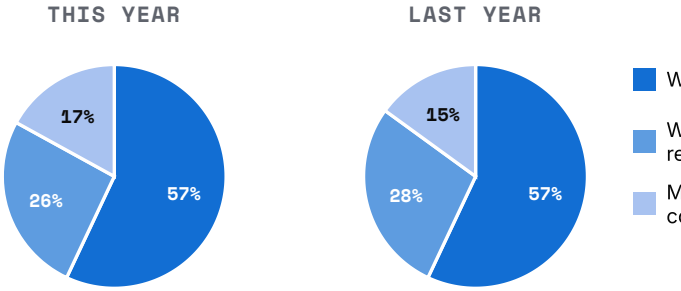
	THIS YR	LAST YR
Enterprise (1,001+)	46%	43%
Mid-market (201–1,000)	26%	27%
SMB (1–200)	28%	29%

Total respondents · This year n=1,026 · Last year n=1,149

FIG. 78

Coding involvement

Do you currently write code or manage software developers as part of your day-to-day responsibilities at work?



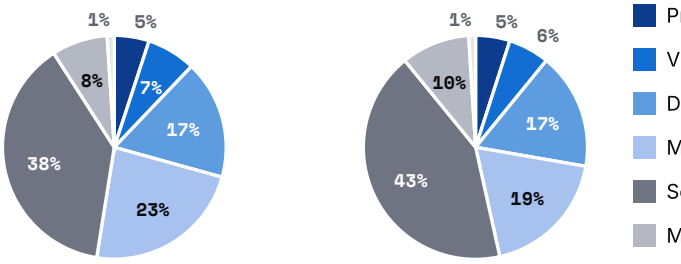
	THIS YR	LAST YR
Write code and manage developers	57%	57%
Writing code is primary responsibility	26%	28%
Manage developers, do not write code	17%	15%

Total respondents · This year n=1,026 · Last year n=1,149

FIG. 79

Seniority

Which of the following best describes your seniority / title? · IC = individual contributor



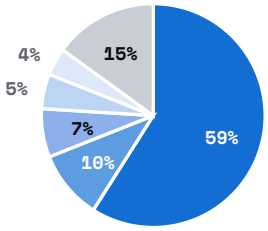
	THIS YR	LAST YR
President or C-level executive	5%	5%
VP or Senior VP	7%	6%
Director or Senior Director	17%	17%
Manager or Senior Manager	23%	19%
Senior, Lead or Principal IC	38%	43%
Mid-level IC	8%	10%
Entry-level IC	1%	1%

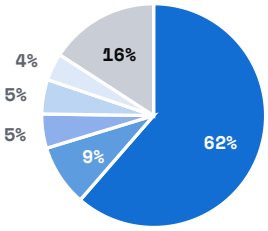
Total respondents · This year n=1,026 · Last year n=1,149 · Figures may not sum to 100% due to rounding

FIG. 80

Industry

In which industry does your organization primarily operate?





	THIS YR	LAST YR
Technology	59%	62%
Financial services & insurance	10%	9%
Healthcare & life sciences	7%	5%
Retail & e-commerce	5%	5%
Manufacturing & automotive	4%	4%
Other	15%	16%

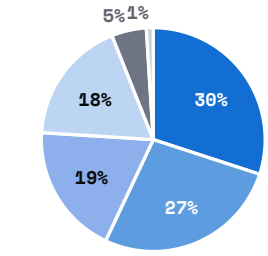
Total respondents · This year n=1,026 · Last year n=1,149

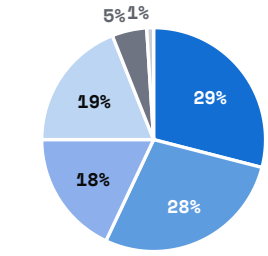
Other: consulting, media, education, energy, telecoms, government, and more

FIG. 81

Primary application type

What is the primary type of application or software you help develop or maintain?





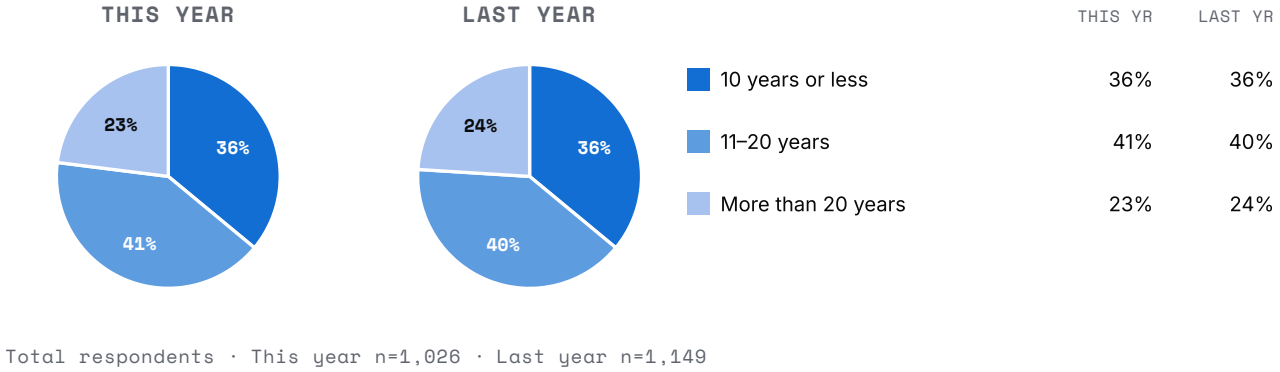
	THIS YR	LAST YR
Commercial applications we sell	30%	29%
Customer-facing web or mobile applications	27%	28%
Internal employee-facing applications	19%	18%
Custom applications built for clients	18%	19%
Applications that run physical equipment	5%	5%
Other	1%	1%

Total respondents · This year n=1,026 · Last year n=1,149

FIG. 82

Experience in a technology role

How many years of professional experience do you have in a technology role?



	THIS YR	LAST YR
10 years or less	36%	36%
11–20 years	41%	40%
More than 20 years	23%	24%

Total respondents · This year n=1,026 · Last year n=1,149

FIG. 83

Analyst's note on data interpretation

Unless noted, results cover all 1,026 respondents; groups under 100 are directional. Percentages are rounded to the nearest whole number, with halves rounded up, and may not total 100% due to rounding or multi-select questions. Agreement combines "agree somewhat" and "agree completely"; positive impact combines "somewhat positive" and "very positive." "Fully trust," "always check," and the SonarQube chapter use only the strongest response. AI's share of committed code and automated-review coverage are averages of respondents' estimates. Before-AI figures rely on recall. ROI and "well-defined" guidelines are self-assessed and were not defined in the survey. Group comparisons show associations, not causation. "X% more likely" uses unrounded percentages. SonarQube user comparisons do not adjust for company size, region, or role.

Glossary of terms

Agentic AI-driven verification	An AI model or agent reviewing code.
Agents, agentic coding	AI that carries out multi-step coding tasks with some autonomy. Used for adoption stages 3 to 5 or where survey questions ask about agents.
AI coding/AI-assisted or generated code	Any use of AI tools or agents to write, change, review, or explain code. The default term in this report.
Deterministic review	The survey category "algorithmic/deterministic/static verification": rules-based analysis that gives the same result every time.
Developers and engineering managers	All respondents (n=1,026), used for combined statements answered in a developer version and a manager version.
Guiderails	Context, standards, and controls (on tokens, permissions, and tool access) that steer agents while they work. Our term, not a survey category.
Harness	The software layer that connects an AI model to the codebase and other tools, and manages the agent's workflow (survey definition).
Open-weight/closed-weight models	Models whose trained weights are, or are not, publicly available to download and modify (survey definition).
Respondents who write code	Respondents for whom writing code is the primary responsibility or who write code and manage developers (n=856).
Software factory	Adoption Stage 5, creating/using software factories (also known as 'dark factories') where agents can work autonomously on tasks.
Verification	Independently checking AI-generated changes for quality and security, inline as an agent writes, at the pull request, or in CI.



STATE OF CODE DEVELOPER SURVEY REPORT | 2027

©2026, SONARSOURCE SÀRL