BLACK DUCK®

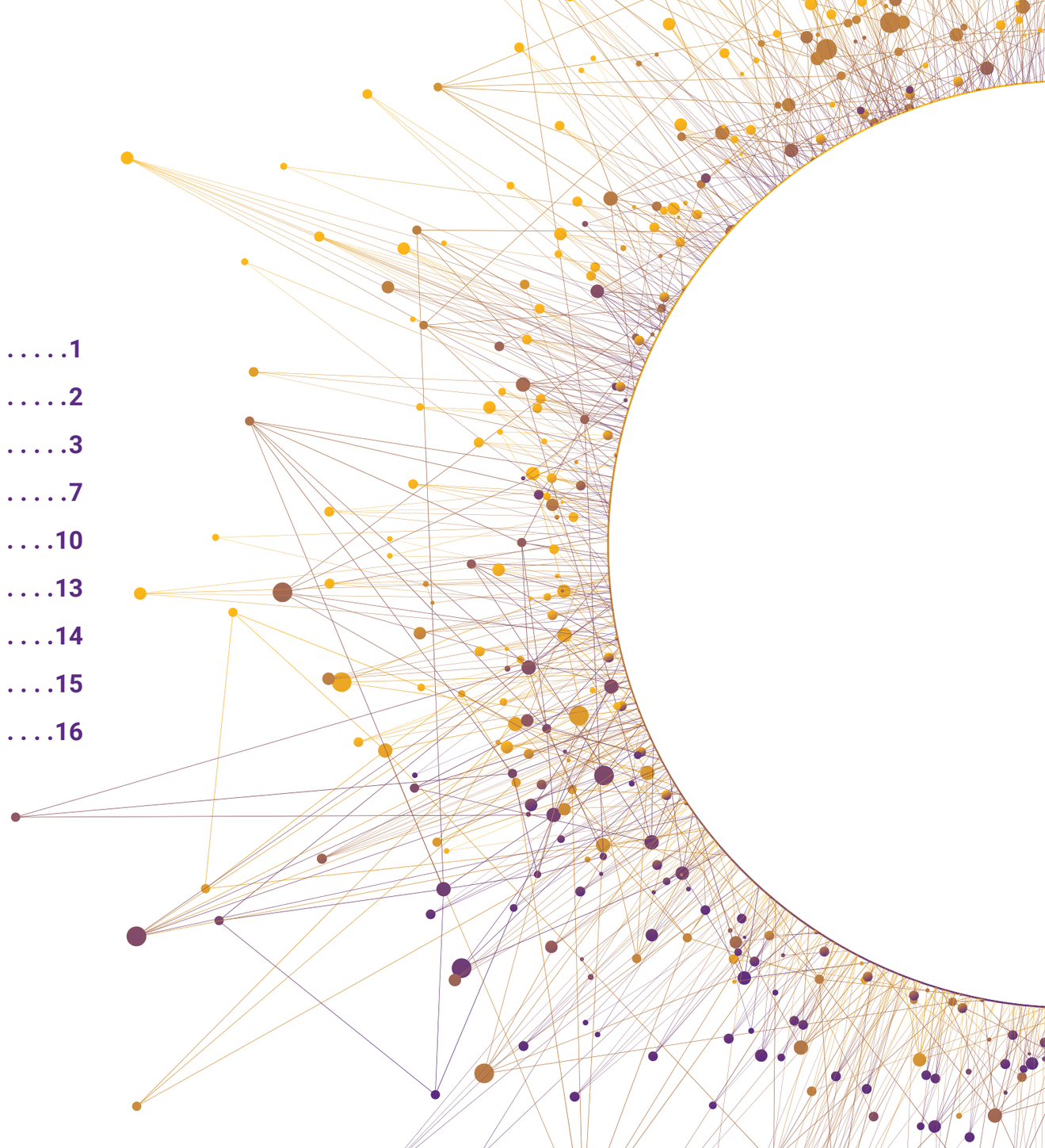
with UserEvidence

# The State of AI-Powered Software Development

## Organizations Need Governance to Unlock AI's True Potential

# Contents





# Executive summary

AI adoption alone no longer creates a competitive advantage. Instead, this edge increasingly comes from how development teams *operationalize* AI.

Many teams report extensive use of AI coding assistants, resulting in notable improvements in velocity and productivity. Yet a lack of governance and limited visibility lead to an array of bottlenecks further down the software development and review process that largely negate these efficiency gains.

Currently, AI simply redistributes development work rather than eliminating it altogether. The next phase of AI maturity must therefore focus on scaling, managing, and auditing AI-assisted code.

In this report, we explore how organizations can benefit from more deliberate implementation of critical processes, effective governance, and specialized tooling. Together, these efforts transform pipelines into the greatest force multiplier for a business, building on the efficiency-based foundation that AI coding assistants have established.

We partnered with third-party research firm UserEvidence to conduct an independent market survey of 831 software engineers and development and operations (DevOps) professionals in March 2026. The research sample was vendor-neutral, but neither Black Duck nor UserEvidence customers were excluded from participating.

# Key findings

## AI provides major efficiency gains and a full day back per week…

AI coding assistants contribute to improved productivity and release velocity for nearly all software development teams (92%), with 58% seeing a major improvement. On average, AI coding assistants save developers eight hours per week.

## …But productivity is offset by workflow trade-offs

Overall, 90% of teams encounter issues with AI-generated code that span the development workflow. The most significant bottlenecks include manual review (52%), security testing (51%), code rework (48%), and prompt iteration (41%).

## AI governance lags behind developer demand for oversight but multiplies AI impact

More than two-thirds of developers (68%) think it's extremely important to have a clear, automated system for tracking AI-generated code. Yet fewer than a third of teams (30%) have full governance in place for AI coding assistant adoption and oversight. Those who do have full governance in place are 55% more likely to see a major improvement to efficiency.

## Concern with AI-assisted code security increases with usage

Nearly two-thirds of teams (64%) express moderate or extreme concern about AI coding assistants introducing security defects or vulnerabilities. Those with extreme concerns tend to be heavy AI users, with 51% of this subgroup using AI coding assistants for the majority of new development.

## Demand grows for AI security agents but not without human oversight

In an ideal automated workflow, more than half of teams (56%) think a dedicated AI security agent should evaluate AI-generated code, yet most (84%) prefer to keep a human in the loop via pull requests or real-time integrated development environment (IDE) suggestions. While developers want automation, pairing it with human oversight allows them to strike a balance between scaling productivity and maintaining security.

**92% of teams see improved productivity and release velocity by using AI coding assistants**

# AI coding assistant adoption drives major productivity gains

As AI coding assistants become increasingly common, most teams see improved productivity, which has resulted in an increase in the volume of code they can deliver. While most are satisfied with the quality, factors like utilization, sophistication, and seniority all affect how teams perceive AI coding outcomes.

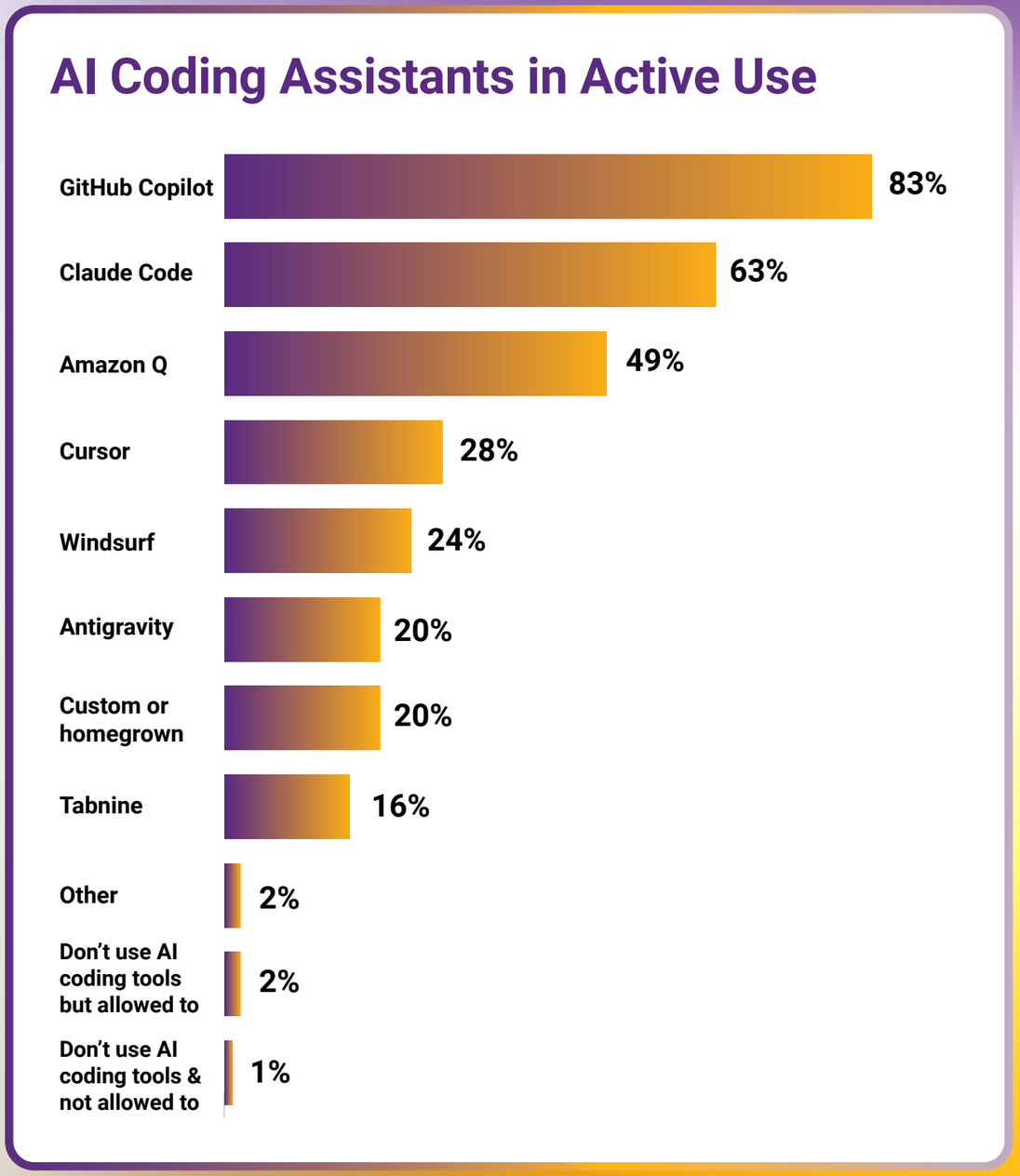
## AI coding tools reach mass adoption and accelerate new development

Nearly all survey respondents (97%) are actively using AI coding assistants in their development environments. Just 2% don't use AI coding assistants, even though they're permitted to, and 1% indicate that their organization doesn't allow AI coding assistants.

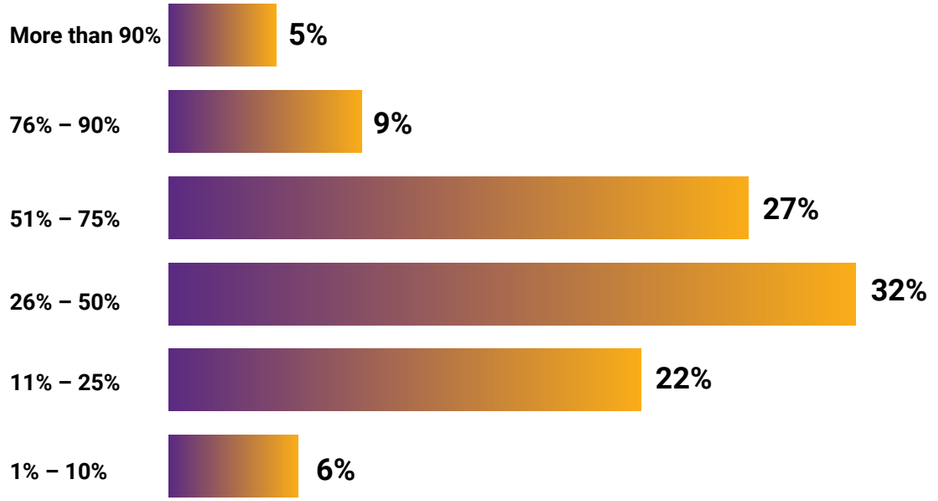
Overall, GitHub Copilot (83%) and Claude Code (63%) are, by far, the most popular. Fewer than 50% of respondents report using any other prebuilt tool, and 20% have developed custom or homegrown tools.

However, the majority (88%) actively use more than one AI coding assistant, indicating that software development teams have moved beyond experimentation and have now identified distinct use cases for different tools.

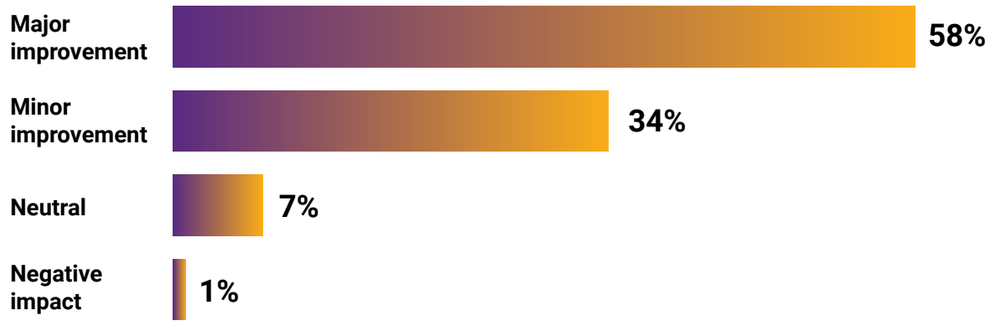
Indeed, most developers have fully incorporated these tools into their workflows. About seven in 10 developers use AI coding assistants for more than 25% of new development efforts. Four in 10 are heavy users, relying on AI assistance for at least 50% of these efforts.



## Percentage of New Development Efforts for Which Developers Use AI Coding Assistants




## Impact of AI Coding Assistants on Productivity and Release Velocity




Although this AI-enabled utilization rate may seem high, additional usage doesn't necessarily imply increased velocity or improved productivity on its own. The AI utilization rate for new development may vary depending on the type of application (e.g., revenue generating, internal business, third party). This is important to consider when evaluating how efficiency and, consequently, risk change based on AI usage.

For example, if an organization is hesitant to use AI for revenue-generating apps because of the business criticality and potential risks, the AI utilization rate could be lower and, therefore, see less of a potential efficiency or productivity gain. The inverse could be true for internal employee apps behind a firewall where there may be a greater AI utilization rate and greater efficiency and productivity gains but less of a potential net impact on the business overall.

## Enthusiastic adopters and senior leadership report outsized AI efficiency gains

Almost all respondents (92%) credit AI coding assistants with an increase in productivity and velocity. More than half (58%) describe the impact as a major improvement, allowing them to release more new features faster than ever before. However, outcomes vary widely depending on output volume, seniority level, and the number of AI coding assistants in use.

It probably comes as no surprise that the most frequent AI users experience the greatest efficiency gains. More than three-quarters of those who use AI coding assistants for at least half of new development (78%) report a major improvement in productivity and velocity, indicating that the impact of AI coding assistants scales with usage.

The teams that see the biggest increase in productivity also tend to use more AI coding assistants, with those who report major improvements using an average of three AI coding assistants, and those who report negative outcomes using just one tool on average. This suggests that the benefits they see with one AI model makes them more amenable to trying others, or that greater benefits may require teams to diversify and find the most relevant or capable tool for their tech stack and development needs.

Respondents' functional roles play an important part in how they perceive productivity as well. In general, respondents at higher seniority levels believe their organization has experienced greater efficiency gains. About three-quarters of C-suite-level respondents (74%) report a major improvement in productivity and velocity, while just 38% of technical contributors report the same outcome, suggesting that senior leaders further removed from day-to-day development may underestimate the friction these tools introduce.

**About three-quarters of C-suite-level respondents (74%) report a major improvement in productivity and velocity, while just 38% of technical contributors report the same outcome, suggesting that senior leaders further removed from day-to-day development may underestimate the friction these tools introduce.**

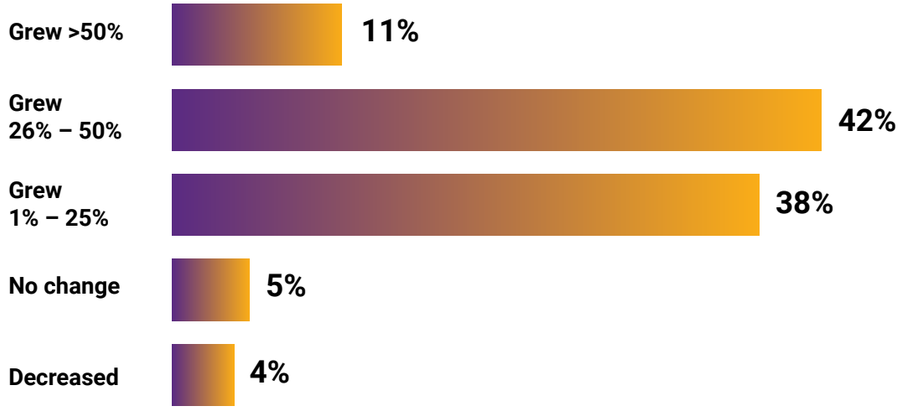
## AI coding assistants increase code volume by more than 25% and save eight hours per week

Along with increasing efficiency, AI assistance has enabled just over half of respondents (53%) to grow code volume by more than 25%. And, for many teams, the more they use AI coding assistants, the more coding volume growth they see.

Eighty-three percent of those who use AI coding assistants for more than half of new development efforts have increased code volume by more than 25%, compared to just 53% of respondents overall experiencing the same growth. What's more, 24% of this high-usage segment has grown code volume by more than 50%, compared to just 11% overall.

In many cases, these tools create substantial time savings for developers, with AI coding assistants saving an average of eight hours per week. Nearly three in 10 developers save at least 11 hours per week.

## Code Volume Growth Due to AI Assistance in the Past 12 Months




AI coding assistants have made it trivial to generate large blocks of code, but higher volume is a double-edged sword. On one hand, it's a massive win for breaking through the "blank page" problem, accelerating boilerplate generation and scaffolding new features. On the other hand, code is often viewed as a liability: stuffing more lines into a repo expands the attack surface, introduces subtle logical vulnerabilities, and creates massive pull request fatigue. Managing this requires moving away from generic "AI governance" and focusing on robust automated guardrails in the CI/CD pipeline.

While efficiency doesn't directly imply quality, most respondents rate the quality of code generated by AI assistants as good (54%) or excellent (27%). These responses vary based on role, though, with senior leadership more likely to put a positive spin on the situation.

Compared to all respondents, those in C-level roles are 78% more likely to consider AI-generated code quality excellent (48% versus 27% overall). In contrast, just 8% of technical contributors and 9% of first-line managers say the same about code quality.

C-level execs may be overestimating the quality that AI coding assistants can deliver on their own. Alternatively, teams may fix the downsides before they ever reach execs, essentially reallocating work to technical team members that does not get recognized or accounted for when the project is viewed in aggregate.

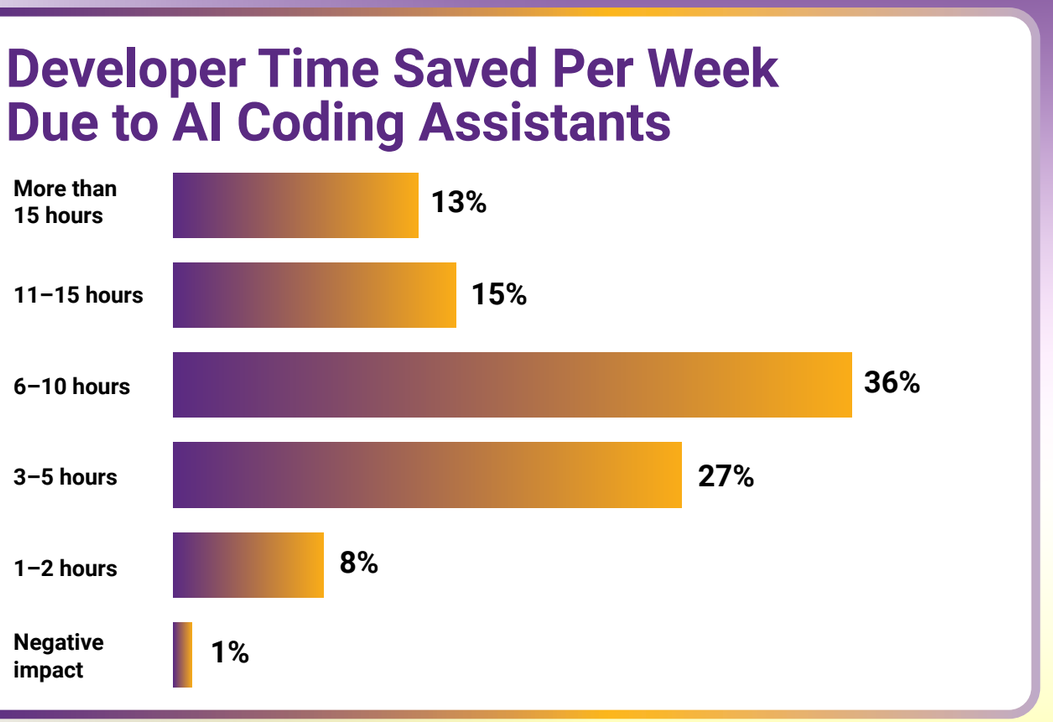
## AI code redistributes work across the pipeline

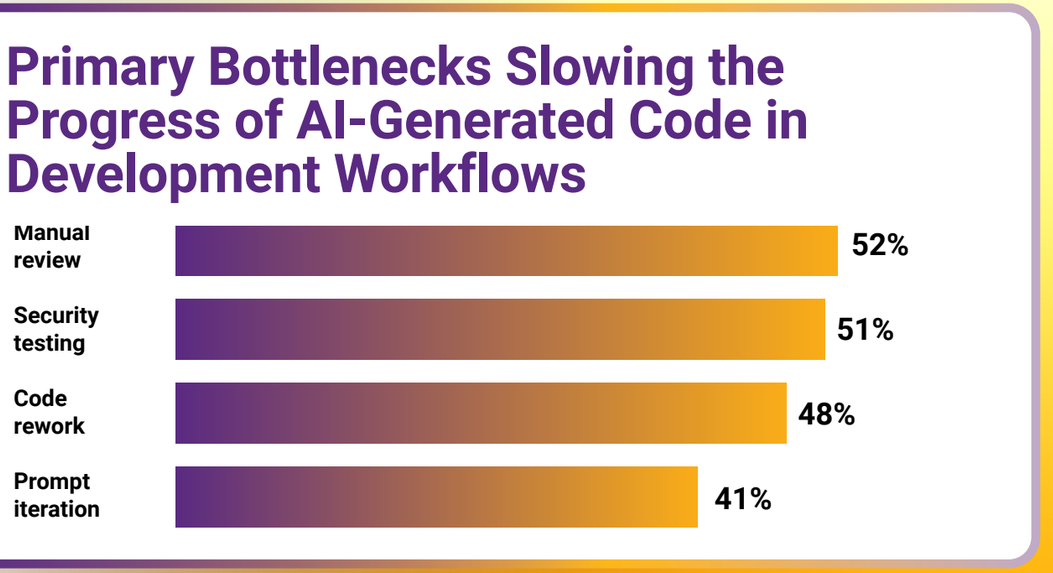
Despite the productivity gains of using AI coding assistants, nearly nine in 10 respondents report ongoing challenges with AI-generated code. The most common bottlenecks include manual reviews, security testing, code rework, and prompt iteration.

These issues indicate that AI coding assistants don't necessarily reduce overall workload, they just move more work downstream, redistributing it to other points in the software development lifecycle (SDLC), such as pre-commit, post-commit, quality assurance (QA), and testing.

As AI-assisted code volume grows, however, it naturally affects certain workflows more than others. For those with AI code volume growth over 50%, the need for additional security testing and vulnerability remediation is the biggest bottleneck, with 57% of respondents pointing to it as an issue.

AI adoption has reached its inflection point, and teams are realizing productivity gains early in the SDLC. With AI coding assistants in the passenger seat, developers are reaching their destination but only by driving faster on a longer road. To achieve real improvements, teams must address lingering questions about quality and the shift of engineering effort downstream.



# Governance is the key to increased AI ROI

While AI coding tools increase efficiency, they also introduce security concerns and operational constraints that limit scalability. Informed, deliberate governance is the key to increased AI return on investment (ROI), establishing guardrails rather than stop-and-go checkpoints.

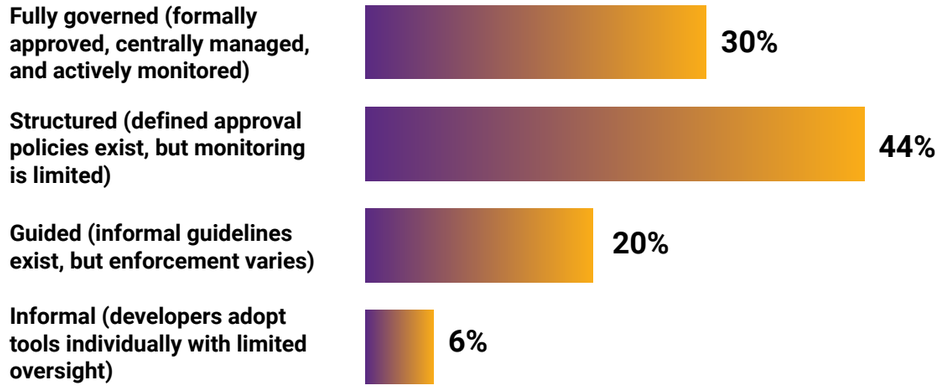
## Governance and process multiply AI returns

Resolving AI coding bottlenecks may not be straightforward, as fewer than a third of respondents (30%) have a fully governed approach to AI coding assistant adoption and oversight. A quarter don't even have defined AI coding policies to guide developers.

However, those who have a fully governed approach to AI code are 55% more likely to see AI coding assistants make a major improvement to efficiency (90% versus 58% overall). When implemented thoughtfully, governance and process become the primary multipliers of AI ROI. As our 2025 report Navigating Software Supply Chain Risk in a Rapid-Release World shows, the more standards and regulations that organizations follow, the faster they fix issues and the more prepared they are to manage risk.

In contrast, organizations that lack established AI governance tend to underperform compared to productivity benchmarks. Fewer than half of teams that aren't fully governed (44%) report a major improvement from AI coding assistants (compared to 58% overall and 90% of those with a fully governed approach).

### Current Approach to AI Coding Assistant Adoption and Oversight




> **Those who have a fully governed approach to AI code are 55% more likely to see AI coding assistants make a major improvement to efficiency (90% versus 58% overall).**

## Guardrails turn AI-generated code into real productivity

Ad hoc AI coding assistant usage likely magnifies the downstream impact. When teams lack clear tools or processes, they can't efficiently handle quality or security issues as code volume increases. Governance establishes guidelines for navigating challenges or inefficiencies, which is particularly beneficial amid handoffs from development to QA to AppSec and back again.

With defined guardrails in place, teams gain the confidence they need to move forward, and as a result, they have to do less manual work.

Handling AI code with thoughtful processes, integration, and automation has positive downstream effects. This approach takes full SDLC efficiency into account instead of simply prioritizing developer efficiency.

## AI-assisted code isn't free of security concerns

In general, AI-generated code is the same as human-generated code, albeit written in greater volumes at higher speeds. Therefore, as with human-generated code, there is an expectation that AI-generated code will contain quality or security issues. Nearly two-thirds of respondents (64%) express moderate or extreme concern about AI coding assistants introducing security defects or vulnerabilities.

The more development teams use AI coding assistants, the more security issues they tend to find. In fact, those with extreme security concerns (25% overall) tend to use AI coding assistants more often, with 51% using AI coding assistants for more than half of new development efforts (versus 41% overall).

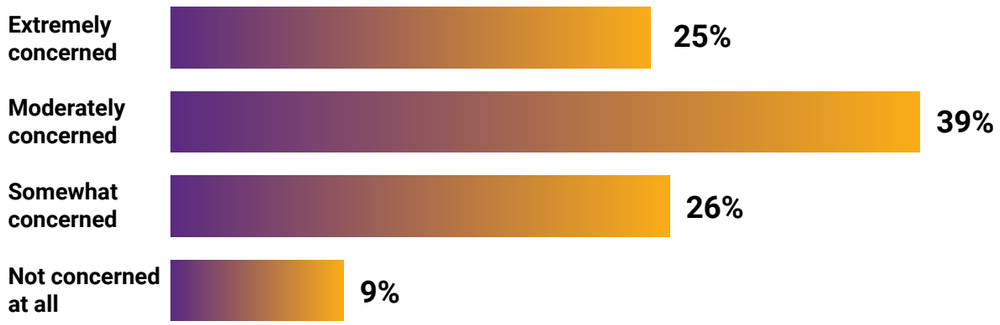
This elevated concern may come from the burden of greater code volume or the abbreviated windows in which to test due to faster-moving, yet potentially ill-equipped, pipelines.

But despite their hesitation, those who express extreme concern with security issues ultimately tend to be more satisfied with AI-enabled output, possibly because their caution drives them to use AI coding assistants more deliberately and to review the output more carefully.

This group was also 12 percentage points more likely to see a major improvement from these tools (70% versus 58% overall) and 17 percentage points more likely to consider the quality of AI-generated code to be excellent (44% versus 27% overall).

**This approach takes full SDLC efficiency into account instead of simply prioritizing developer efficiency.**

### Level of Concern with AI Coding Assistants Introducing Security Defects or Vulnerabilities



## AI code tracking remains largely manual, even with 68% preferring automation

Overall, most respondents want more structured AI governance. Two-thirds of developers (68%) think it's extremely important to have a clear, automated system for tracking AI-generated code and measuring its impact for debugging, security, and accountability.

However, structural friction often leads developers to settle for manual rather than automated code tracking and impact measurement. While 40% use automated tagging and metadata within their IDE or repository to identify and track any development work that leveraged an AI coding assistant, 38% continue to rely on manual developer comments or documentation in pull requests to do so.

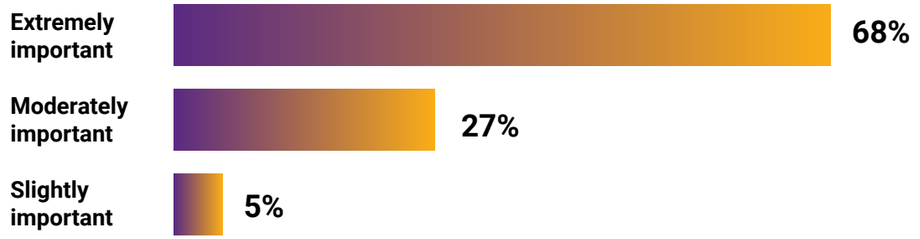
While nearly all teams see growing code volumes, without adequate tagging and tracking of AI-generated code, a large plurality will lose visibility into the structure and contents of that code. The natural consequence is longer investigations into the source of issues detected downstream and extended periods of rework, which itself might add another cycle of inflated code for review.

Many organizations that classify their current approach to AI oversight as structured (43%) or fully governed (17%) rely on manual developer pull request comments. While these teams may *think* they have true governance in place, they may simply be operating with a stated policy that lacks the essential guardrails for improving security and efficiency.

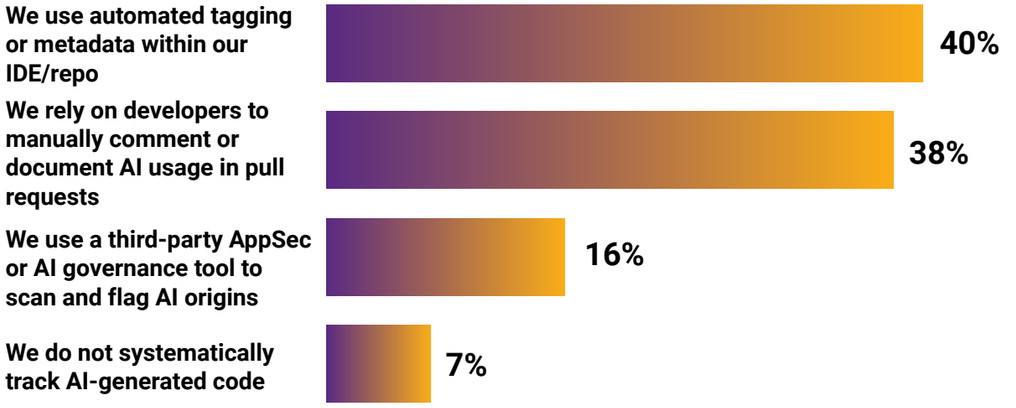
In many cases, establishing clear processes for managing AI-generated code may come down to leveraging the heightened influence of security concerns. Teams that align with security and set automated guardrails can truly capture AI ROI across the entire SDLC, rather than just in the coding phase.

> **Teams that align with security and set automated guardrails can truly capture AI ROI across the entire SDLC, rather than just in the coding phase.**

### Level of Importance for Having an Automated System to Track and Measure AI-Generated Code




### Process for Tracking AI-Generated vs. Human Code



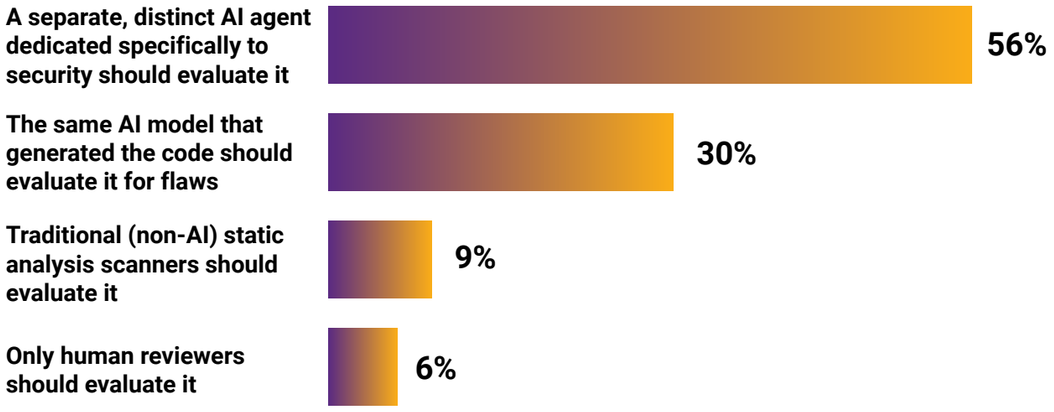
# The future of AI-powered development hinges on security

While most teams are open to AI-powered security tooling, they want to keep a human in the loop. Accordingly, future-focused teams should leverage AI-enabled efficiency gains to allow human developers to manage governance, tracking, and validation of AI-assisted code. Only when someone knows the structure of the code and the source of vulnerabilities can they share the benefits of AI-assisted coding with security teams without completely subsuming those extra eight hours into arduous code review.

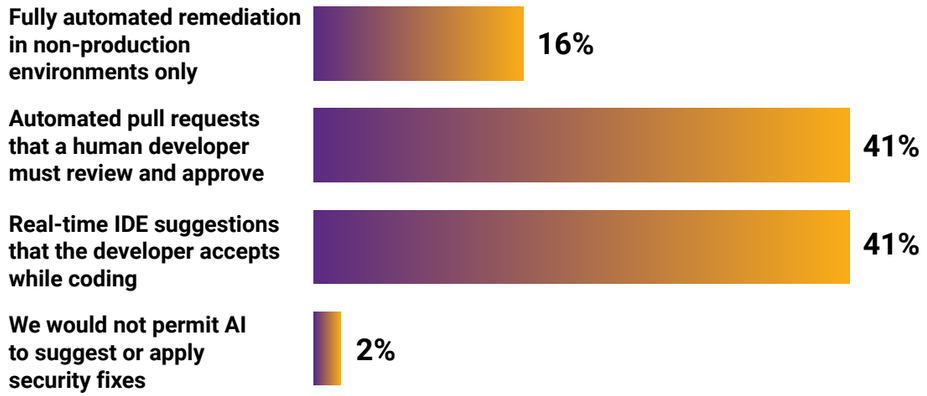
## Teams prefer a dedicated AI security agent to evaluate AI code

Most respondents favor an AI-assisted approach to security, with 86% believing an AI security agent or model should evaluate AI-generated code, yet there's no clear consensus on how to incorporate AI into security testing. While more than half (56%) would prefer a separate AI agent dedicated specifically to security, about a third (30%) think the same AI model that generated the code should also evaluate it.

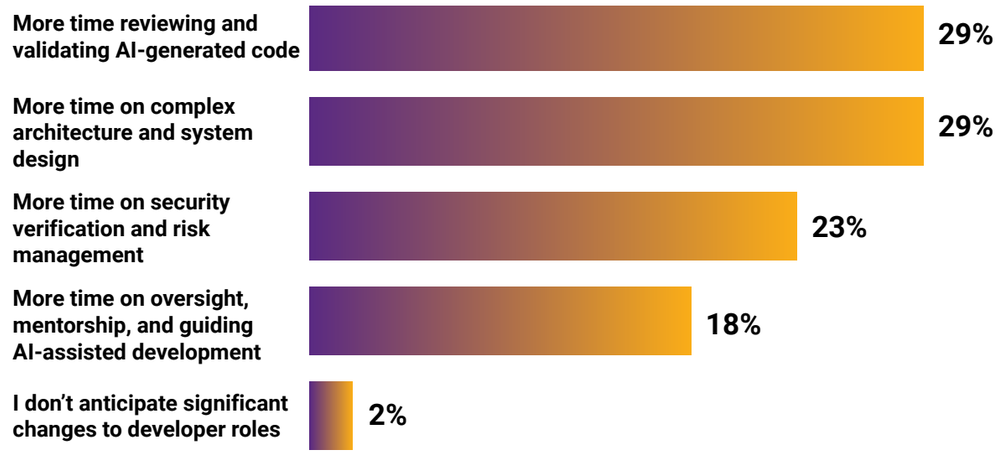
### Ideal Approach to Evaluate AI-Generated Code for Security Flaws



## Preferred Way to Integrate an Autonomous AI Security Agent




## Anticipated Changes to Developer Roles Due to AI in the Next Two Years




## Even with AI, developers anticipate retaining decision-making roles

Organizations are divided on how to integrate an AI security agent into their software development pipeline, with 41% preferring automated pull requests that a human developer must review and approve and another 41% preferring real-time IDE suggestions for developers to accept while coding.

Those on either end of the spectrum are in the minority: only 16% prefer fully automated remediation, and just 2% wouldn't permit AI security agents at all. In other words, developers want AI-assisted security input, but they want to remain in control of the final decision.

Most teams expect that having a human in the loop will continue to be necessary in the near future, with respondents anticipating that developers will spend more time in three key areas: reviewing and validating AI code (29%), complex architecture and system design (29%), and security verification and risk management (23%).

While developers increasingly rely on AI assistants to generate code, they expect to retain their decision-making roles in the software development workflow. As AI handles code production, it reallocates the work, causing developers to shift toward review, validation, and system design.

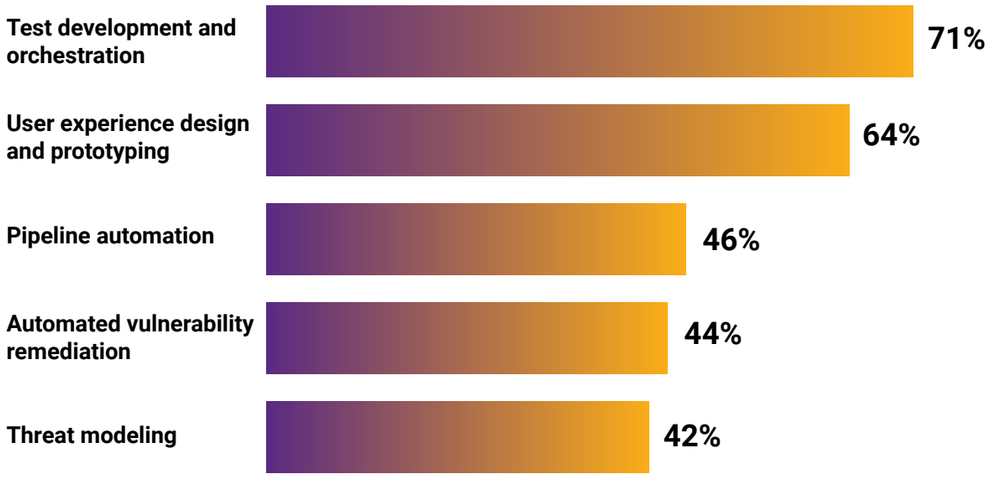
## Future-focused teams need a proactive approach to AI security

In the next 12 months, software development teams are looking to invest in a range of AI-assisted workflows. The most common include test development and orchestration (71%) and UX design and prototyping (64%).

Although these elements may improve efficiency in their respective areas, they're less consequential to the overall pipeline or to the risk posture of associated projects. Ultimately, teams are apprehensive about trusting AI with tasks that involve security or where issues (and the consequences of failing to address them) can propagate.

The software development teams that get ahead understand that, in the near term, AI shouldn't eliminate developers. Instead, it should provide valuable time for teams and companies to invest into other priority initiatives.

### AI-Assisted Development Workflows Under Consideration in the Next 12 Months



# Conclusion: AI impact scales with discipline and governance

AI provides benefits like faster innovation, abbreviated release timeframes, and the potential for improved issue remediation, but these benefits shift engineering effort instead of eliminating it.

The teams that understand how to operationalize AI will win. To fully realize these gains, organizations should focus on guardrails and alignment to standards that serve as the blueprint for faster operations and remediation workflows. When workloads shift from development to QA, DevOps, and AppSec, those teams will be more prepared to perform faster and at scale as well to avoid negating the efficiency benefits.

If AI becomes as adept at finding and exploiting vulnerabilities as early Mythos reports suggest, it's clear that teams must get ahead of three key imperatives:

1

## Maximize the coverage and automation of your application security program

To safely support the sheer volume of AI-generated code, your security testing must operate seamlessly across the entire CI/CD pipeline. Focus on automating project onboarding and running concurrent testing methodologies to ensure comprehensive risk coverage without creating development bottlenecks.

2

## Ensure you can detect and rapidly respond to supply chain vulnerabilities

To mitigate complex supply chain risks and to ensure compliance with regulations like the EU Cyber Resilience Act (CRA), organizations must maintain dynamic and comprehensive Software Bills of Materials (SBOMs) for the software they create and that which they ingest. Prioritize deep vulnerability intelligence, software license compliance, and automated alerting so teams can accurately track, deliberately prioritize, and readily fix emergent risks.

3

## Begin the transition from script-based DevSecOps to a secure agentic SDLC

Move beyond reactive security gates by embedding intelligent, context-aware security agents directly into the developers' preferred coding environment. This approach leverages AI to build application security testing that automatically adapts and responds to events at machine speed.

# Methodology and demographics

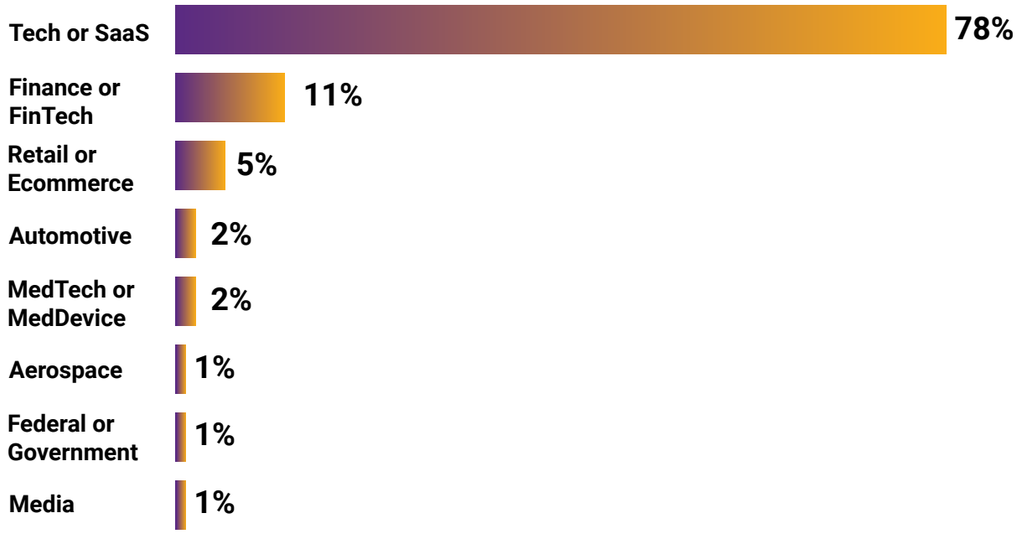
All survey respondents worked for an organization with at least 500 employees. Most were from companies with at least 2,000 employees (67%), and a third had more than 5,000 employees (35%). The average organization size was 3,250 employees.

While these organizations represented a range of industries, the largest segment operated in technology or software-as-a-service (78%). Other industries included finance or financial technology (FinTech) (11%) and retail or eCommerce (5%).

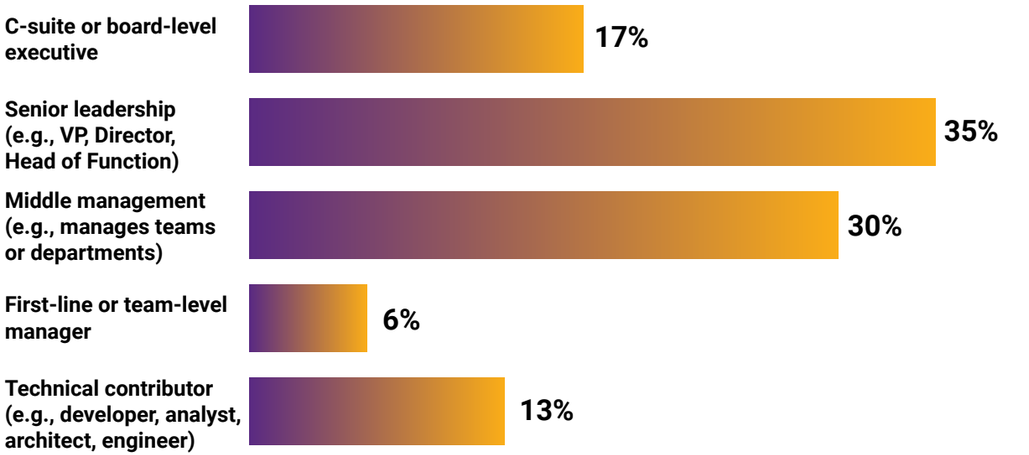
In terms of functional areas, about three-quarters of respondents (77%) primarily worked in application development and software engineering, followed by DevOps and platform engineering (21%).

More than half of respondents were C-suite level or senior leadership (52%). Others were middle management (30%), technical contributors (13%), or first-line managers (6%).

## Respondents' Industries




## Respondents' Professional Levels



# About Black Duck

Black Duck meets the board-level risks of modern software with True Scale Application Security, ensuring uncompromised trust in software for the regulated, AI-powered world. Only Black Duck solutions free organizations from trade-offs between speed, accuracy, and compliance at scale while eliminating security, regulatory, and licensing risks.

Whether in the cloud or on premises, Black Duck is the only choice for securing mission-critical software everywhere code happens. With Black Duck, security leaders can make smarter decisions and unleash business innovation with confidence. Learn more at www.blackduck.com.

**Speak with a Black Duck expert today and find out how you can accelerate security to speeds commensurate with AI-enabled development.**

## Polaris

The Black Duck Polaris™ platform provides a highly scalable, unified AppSec solution designed for the speed of AI-powered software development. Polaris seamlessly integrates across CI/CD pipelines, with automated project onboarding that instantly applies the right security controls. With SAST, SCA and DAST analysis running concurrently, Polaris ensures comprehensive coverage without sacrificing development velocity.

## Black Duck SCA

Black Duck® SCA is a comprehensive supply chain risk management solution that detects and identifies third-party libraries within applications, helping organizations to mitigate associated security and license risks and to sustain the transparency necessary to comply with regulations like the EU CRA. Deliberately automate based on flexible, highly specific policies that balance projects' unique risk tolerances with release speed and readily alert contributors and downstream consumers to emergent vulnerabilities impacting previously scanned projects.

## Signal

Black Duck Signal™ is an agentic application security solution purpose-built for the era of AI-driven software development. It combines the generative and autonomous capabilities of large language models with over 20 years of human-curated security intelligence—including vulnerability data, coding patterns, and best practices—to power intelligent test-and-remediate workflows.

Signal helps development teams prevent, detect, and fix vulnerabilities and software supply chain risks at machine speed. By integrating directly into AI coding assistants and agentic development lifecycles, it delivers autonomous application security with the deterministic results, scalability, and cost efficiency required to secure modern software at scale.

# About UserEvidence

UserEvidence is a software company and independent research partner that helps B2B technology companies produce original research content from practitioners in their industry. All research completed by UserEvidence is verified and authentic according to their research principles: identity verification, significance and representation, quality and independence, and transparency. All UserEvidence research is based on real user feedback without interference, bias, or spin from our clients.

## UserEvidence research principles

These principles guide all research efforts at UserEvidence—whether working with a vendor's users for our Customer Evidence offering or industry practitioners in a specific field for our Research Content offering. The goal of these principles is to give buyers trust and confidence that you are viewing authentic and verified research based on real user feedback, without interference, bias, and spin from the vendor.

### 1 Identity verification

In every study we conduct, UserEvidence independently verifies that a participant in our research study is a real user of a vendor (in the case of Customer Evidence) or an industry practitioner (in the case of Research Content). We use a variety of human and algorithmic verification mechanisms, including corporate email domain verification (i.e., so a vendor can't just create 17 Gmail addresses that all give positive reviews) and pattern-based bot and AI deflection.

### 2 Significance and representation

UserEvidence believes trust is built by showing an honest and complete representation of the success (or lack thereof) of users. We pursue statistical significance in our research and substantiate our findings with a large and representative set of user responses to create more confidence in our analysis. We aim to canvas a diverse swath of users across industries, seniorities, personas—to provide the whole picture of usage, and allow buyers to find relevant data from other users in their segment, not just a handful of vendor-curated happy customers.

### 3 Quality and independence

UserEvidence is committed to producing quality and independent research at all times. This starts at the beginning of the research process with survey and questionnaire design to drive accurate and substantive responses. We aim to reduce bias in our study design and use large sample sizes of respondents where possible. While UserEvidence is compensated by the vendor for conducting the research, trust is our business and our priority, and we do not allow vendors to change, influence, or misrepresent the results (even if they are unfavorable) at any time.

### 4 Transparency

We believe research should not be done in a black box. For transparency, all UserEvidence research includes the statistical N (number of respondents), and buyers can explore the underlying blinded (de-identified) raw data and responses associated with any statistic, chart, or study. UserEvidence provides clear citation guidelines for clients when leveraging research that includes guidelines on sharing research methodology and sample size.


BLACK DUCK®