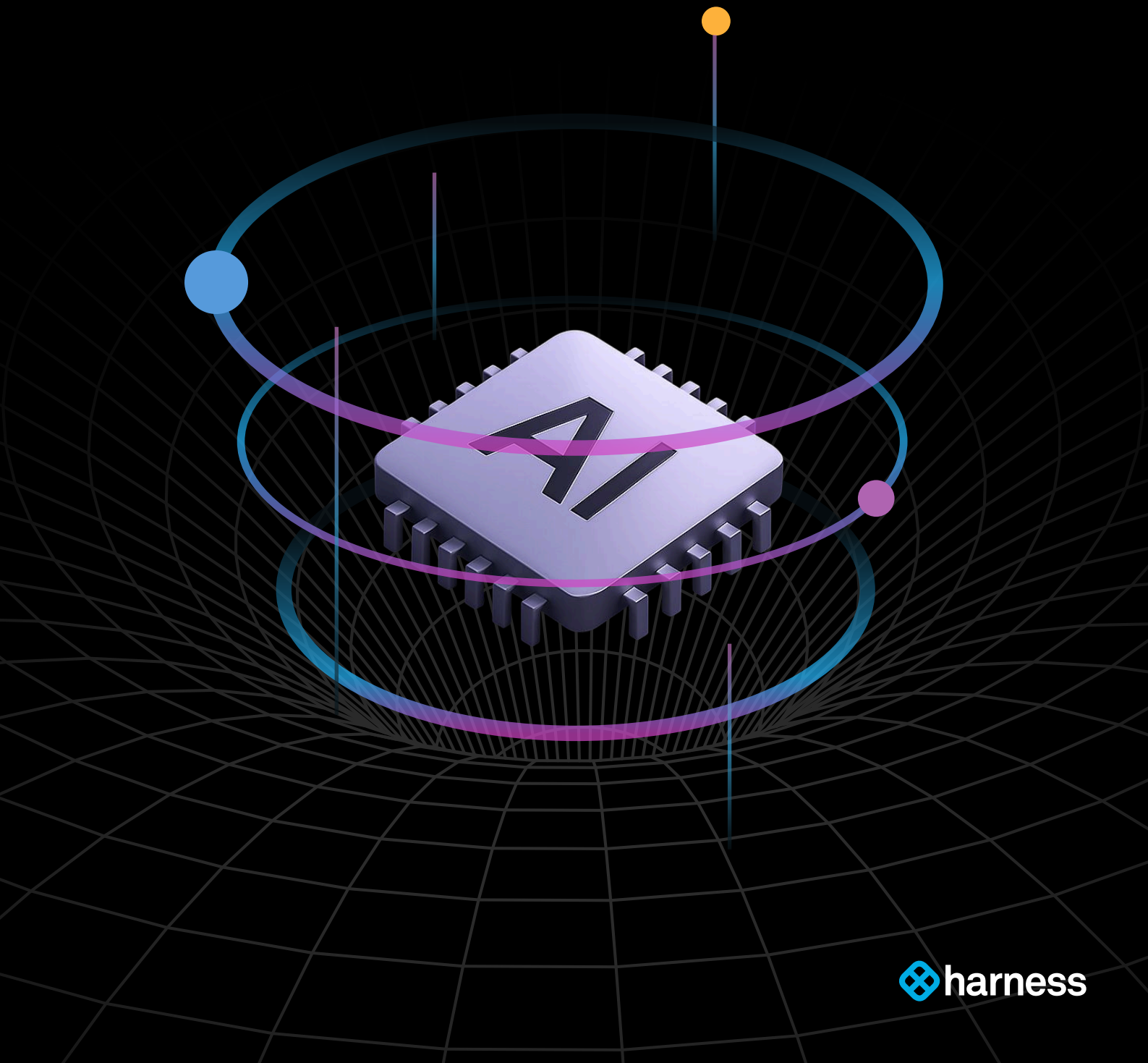
THE STATE OF

AI in Software Engineering 2025



harness

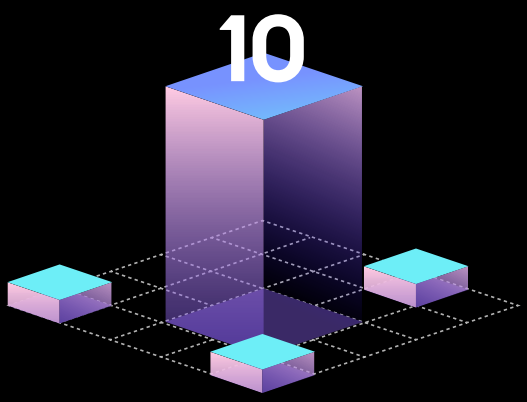
Executive Summary

AI-assisted coding has gone from an experimental edge to an everyday reality. For engineering teams, the promise is immense: faster development cycles, improved code quality, and more time for innovation. In our survey of 900 engineers, platform leaders, and technical managers across the United States, the United Kingdom, France, and Germany, the majority report that the use of AI assistants is leading to faster shipping speeds.

Yet, beneath the surface-level optimism lies a hard truth: AI's speed boost is uneven. The benefits gained in code creation are being throttled by processes further down the software delivery lifecycle (SDLC). Downstream processes, including testing, deployment, security, and compliance, have not matured at the same pace, creating the **AI Velocity Paradox.**

The result is a dangerous operational blind spot. Organizations believe they are accelerating when in reality, they are simply pushing unverified code into production faster.

Key Findings Include



1.

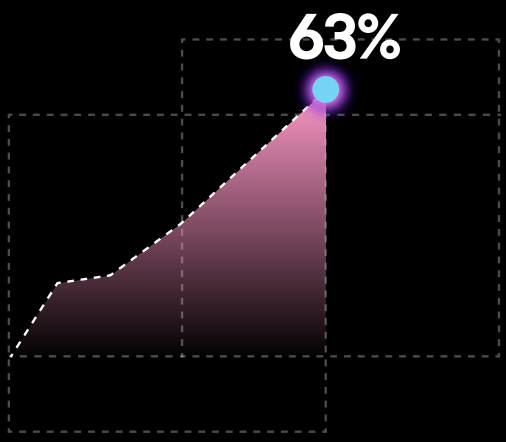
Pervasive AI Adoption

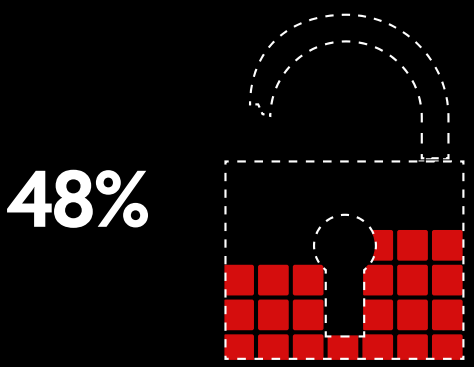
On average, development and engineering teams use between **eight to ten distinct AI tools**, while more than a third (36%) use an even wider array.

2.

A Clear Velocity Lift

Since adopting AI, **63% of organizations are shipping code to production faster.**





3.

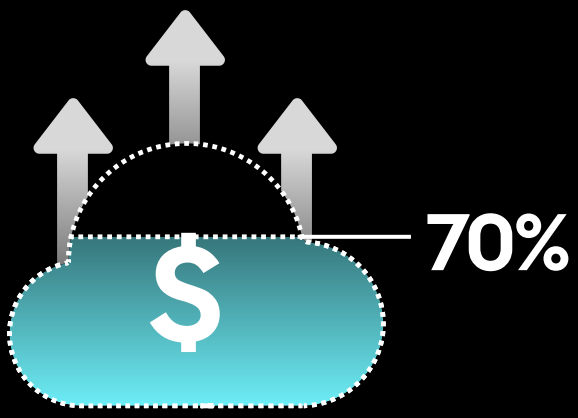
Increased Security and Quality Risk

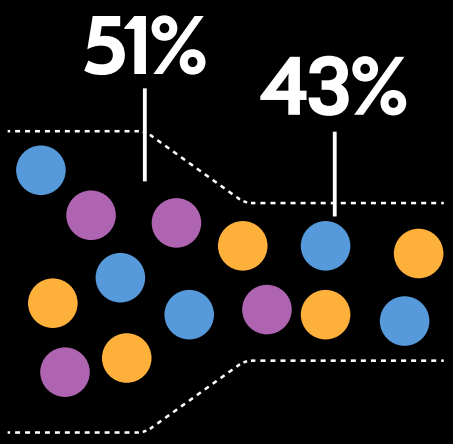
Nearly half (48%) of teams are concerned they will see an increase in software vulnerabilities due to their use of AI-coding assistants, and **45% of all deployments linked to AI-generated code lead to problems.**

4.

Rising Costs

More than two-thirds (70%) of organizations are concerned that AI assistants could cause cloud costs to spiral out of control, because it is so easy to deploy inefficient code.





5.

The Downstream Bottleneck

The proportion of coding workflows that are automated sits at 51%, but is just 43% for CI/build pipeline creation and execution.

"Organizations that fail to integrate AI safely and securely across their SDLC in the next 12 months will go the same way as the dinosaurs."

- say 74% of respondents

1. The New Normal: Pervasive AI Adoption and Tool Sprawl

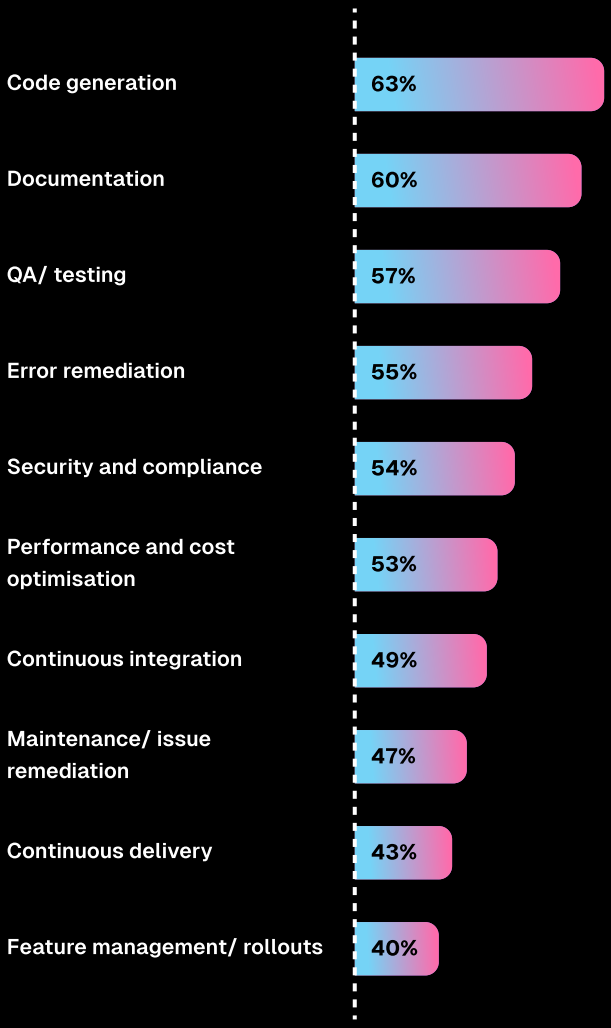
The adoption curve for AI in software engineering isn't just steep, it's vertical. AI is no longer a sidecar; it's part of the engine.

Organizations have already adopted AI in an array of use cases across the SDLC, and expect to see significant efficiency gains. However, the use of AI in downstream stages of delivery lags behind its application for code generation, which threatens to hinder organizations' ambitions.

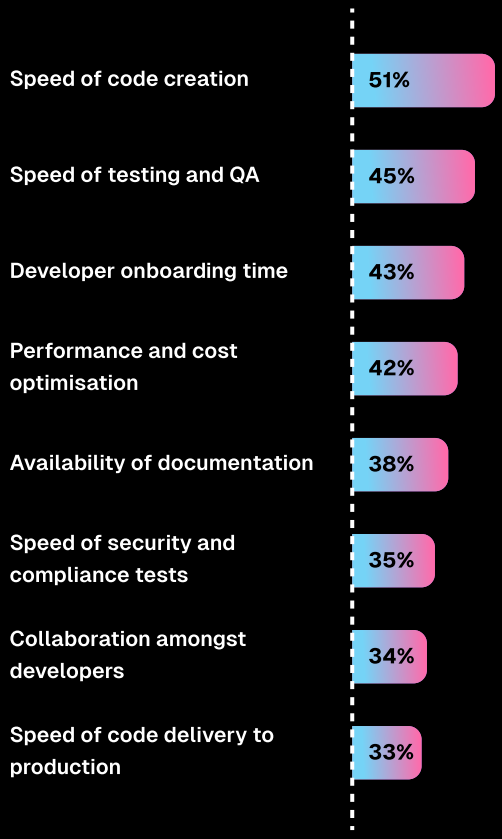
"Within the next five years, software delivery will be dominated by AI agents working alongside human engineers"

- say 80% of development teams and their leaders

Areas of software delivery where organizations have already adopted AI:



Aspects of software delivery that respondents expect to see significant impact from AI.



Tool proliferation

On average, organizations use **eight to ten distinct AI tools** for software engineering, and more than a third (36%) are juggling an even greater number. This paints a picture not just of AI abundance, but of significant tool sprawl. While each new tool may add marginal value, it also introduces undeniable complexity and lengthens developer onboarding time.

the average time it takes to fully onboard a new software engineer

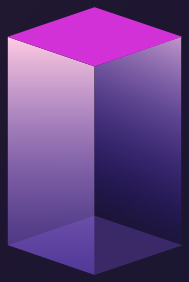
“Constant context switching is mentally draining and kills developer productivity”

- say 71% of respondents

Managing security, ensuring compliance, and monitoring performance across a fragmented toolchain becomes a significant governance headache; one that most organizations are only beginning to confront. This is the first crack in the foundation of AI-driven development, setting the stage for bottlenecks to come.

2. The AI Velocity Paradox: When Speed Creates Drag

When asked about the impact of AI coding tools, 63% of respondents say their developers ship code to production more frequently.



63%

However, upstream speed is only one part of the equation. Faster code creation does not guarantee overall development velocity.

> **"The use of AI coding assistants is like squeezing a balloon – the volume of work stays the same, it's just forced from one side to another."**
>
> - say 67% of development teams and their leaders

The speed boost from AI-assisted coding is creating a pressure wave that is crashing against a wall of under-automated, legacy downstream processes. While developers are writing code faster than ever, the systems meant to test, secure, and deploy that code are struggling to keep up. This has led to the emergence of the AI Velocity Paradox.

Limited Downstream Automation

There is a stark imbalance in automation maturity across the SDLC. While 51% of coding workflows are automated on average, organizations' use of these capabilities drops off consistently from there.

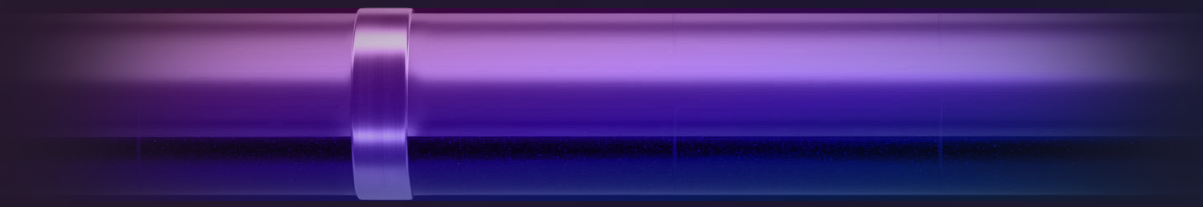
51% Coding/ Development

48% Developer environment creation

47% Quality and resilience testing/ QA

46% Security and compliance testing

46% Documentation



45% Continuous delivery

45% Incident management

43% Continuous Integration/Build pipeline creation and execution

43% Performance and cost optimization

38% Feature Management/ Rollouts

The so far limited use of AI in Continuous Delivery (CD) is of particular concern.

Only 6% of respondents characterize their CD process as fully automated.

The majority (85%) say their process is somewhere between 1-75% automated, highlighting that it remains very normal for significant manual effort to be required in each release.

Significant Manual Effort

2x velocity gain

What's more, for organizations with less than a quarter of their CD workflows automated, only 26% have seen an increase in the frequency at which code is shipped to production from their use of AI coding tools. This jumps to 57% in those that have between one and three-quarters of CD processes automated. Moving from low to moderate automation in CD, therefore, more than doubles the likelihood that organizations will see a velocity gain from AI coding tools.

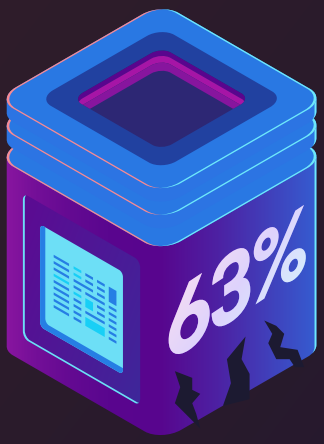
Cracks Beginning to Emerge

The gap between automated code creation and validation is a primary source of risk, quality issues, and hidden costs that can stall the overall velocity of software delivery. Only 41% of respondents are fully confident that their deployment checks and governance processes will always be able to prevent bugs from AI-generated code getting into production.

"It's impossible to strike the perfect balance between delivering code faster and ensuring its reliability and security"

- say two-thirds (66%) of respondents

This challenge is only set to worsen as vibe coding continues to take off amongst non-specialist or less experienced developers.



Nearly two-thirds (63%) of respondents say vibe coding is “a disaster waiting to happen, and skilled developers are about to be inundated with requests to fix other people's shoddy work.”

Organizations are accelerating directly into a new set of bottlenecks. The time saved by developers will be paid for downstream by operations, security, and finance teams. This is the drag that turns a localized speed boost into a system-wide slowdown.

“Without automated guardrails, organizations are going to face some very painful and costly mistakes from non-technical teams trying their hand at vibe coding”

- say 78% of respondents

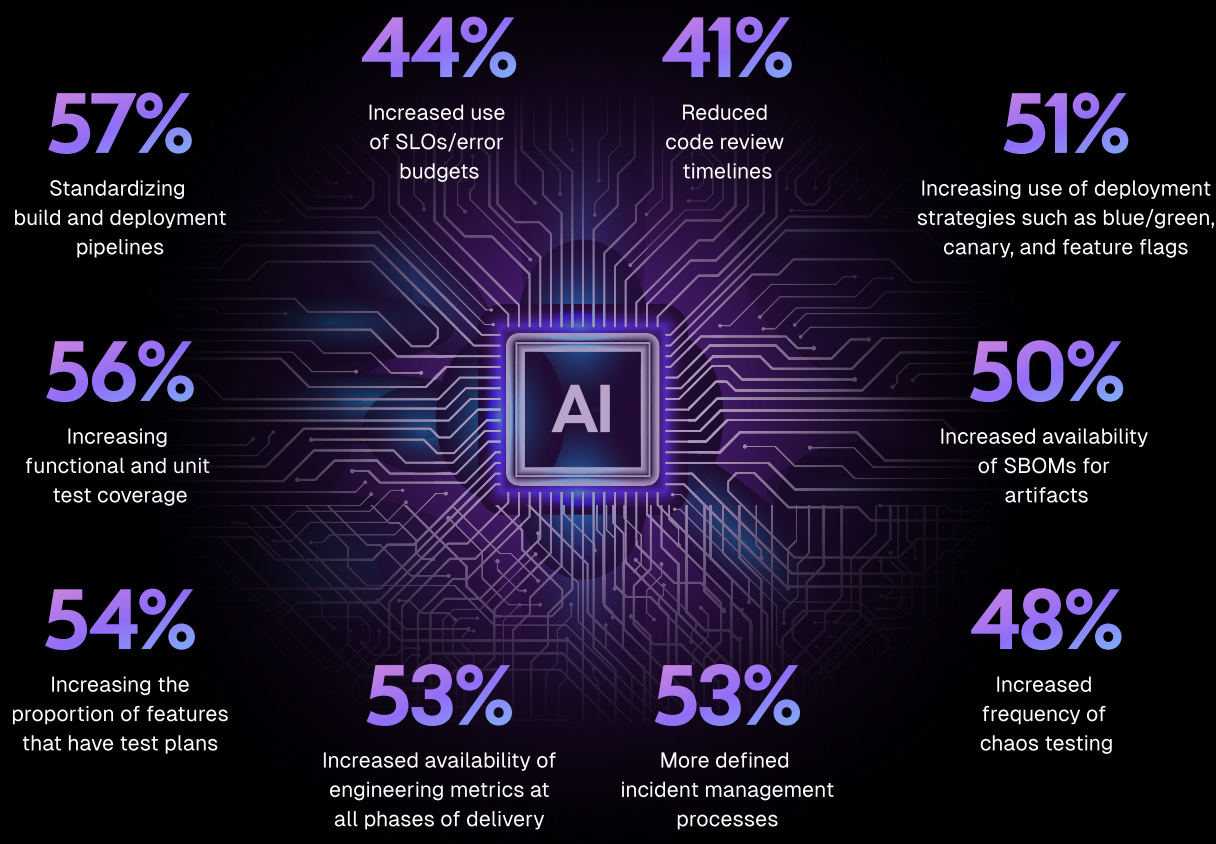
3. Downstream Bottlenecks Slow AI-Powered Delivery

The drag on velocity isn't caused by a single point of failure, but by a series of interconnected bottlenecks across the downstream phases of the SDLC. When we examine the specific areas of friction, several patterns emerge.

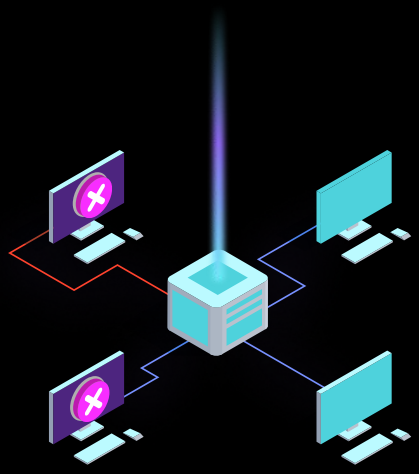
DevOps and Pipeline Proliferation

Creating, securing, and maintaining a unique delivery pipeline for every new service is simply not scalable at the speed of AI-driven development. Perhaps in light of this, guardrails for standardizing build and deployment processes emerged as the most important priority for increasing the use of AI in software delivery.

Organizations that rate automatically enforced guardrails and policies in the following areas as very important to increasing their use of AI across the SDLC:



More Deployment Failures With a Wider Fallout

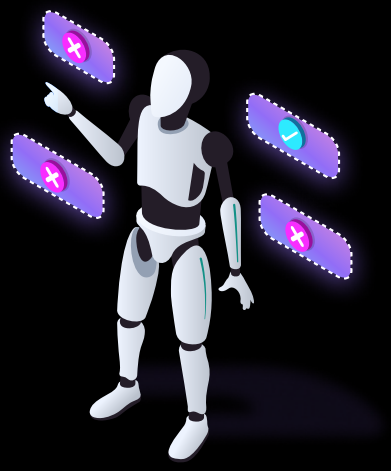


45%

of deployments involving AI-generated code introduce problems.

72%

of organizations have already experienced at least one production incident directly caused by AI-generated code.



Organizations remain relatively immature in their use of feature management to mitigate the risk of releases for code created with AI. Less than half say their developers are using feature flags for many of the primary use cases that help to prevent coding errors from creating a widescale production incident.

Use cases where developers use feature flags for AI-assisted/generated code.

Use case

58%
Kill switches and rollbacks

50%
A/B testing and experimentation

47%
Gradual rollouts/ Phased releases

43%
Controlled access to new features

31%
Testing features in production

"If they aren't managed carefully, AI assistants could significantly widen the blast radius of failed software releases, creating more customer disruption, business risk, and developer stress,"

- say 73% of respondents

Increased Manual Toil and Risk

Any time saved during code creation is being reallocated to cleanup during downstream stages of delivery. One third **(33%)** of organizations are concerned that the use of AI-assisted coding tools will lead to an increase in the amount of manual downstream work such as QA, testing, and integration. Whether this work falls to a dedicated team or back to the developer, it represents a direct tax on innovation.

In particular, faster code generation threatens to overwhelm teams who rely on manual security reviews. Nearly half **(48%)** of organizations are concerned they will see an increase in software vulnerabilities and security incidents as a result of their adoption of AI-assisted coding tools, and **43%** worry they will face increased regulatory non-compliance.

Unexpected Cost Overruns

Inefficient, AI-generated code can also have a direct impact on the bottom line. More than two-thirds **(70%)** of respondents are concerned that their cloud costs could spiral out of control as the use of AI assistants increases, because it's so easy to deploy inefficient code.

These bottlenecks demonstrate that true velocity is a measure of the entire system, not just its fastest component. Addressing these downstream friction points is the only way to realize the full promise of AI in software development.

> **"The use of AI needs to extend across the entire SDLC if we're to realise the true potential of coding assistants for accelerating delivery,"**
>
> - say 83% of respondents

4. Navigating the Paradox: The Four Quadrants of AI Maturity

The AI Velocity Paradox forces organizations into one of four distinct quadrants, defined by their maturity in both upstream AI adoption (coding) and downstream AI-powered automation (testing, security, deployment).

	Low Downstream Automation	High Downstream Automation
High AI Adoption (Coding)	The Danger Zone Fast, but Fragile	The Velocity Leaders Fast and Resilient
Low AI Adoption (Coding)	The Laggards Slow and Fragile	The Cautious Planners Slow, but Stable

The Danger Zone: High AI Adoption, Low Downstream Automation

This is where the paradox lives and where most organizations find themselves today. They have enthusiastically adopted AI for code generation, achieving the initial speed boost. However, their downstream processes remain manual and brittle. They are, in effect, running face-first into a wall. The result is a high-risk environment characterized by:

Frequent deployment failures

Increased security vulnerabilities

Rising manual toil for QA and Ops teams

Unpredictable costs

The Laggards: Low AI Adoption, Low Downstream Automation

These organizations have not embraced AI for coding and still rely on manual processes throughout the software delivery lifecycle. They are slow, inefficient, and falling behind the competition. While they may not experience the acute pain of the Velocity Paradox, they are failing to capitalize on the single biggest technology shift in modern software development.

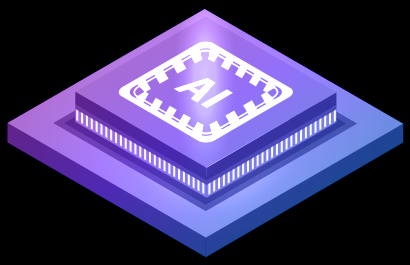
The Cautious Planners: Low AI Adoption, High Downstream Automation

A small and disciplined group, these organizations have mature, automated pipelines but have been slow to adopt AI for coding. They are stable and reliable, but risk being outpaced by more agile competitors. Their robust downstream foundation, however, positions them to become Velocity Leaders as soon as they choose to accelerate their upstream AI adoption.

The Velocity Leaders: High AI Adoption, High Downstream Automation

This is the target state. Velocity Leaders have embraced AI not just for coding, but across the entire software delivery lifecycle. They use AI-powered automation to test, secure, deploy, and verify code, ensuring that speed is matched with safety and resilience. For these organizations, AI is a true competitive advantage, enabling them to deliver value to customers faster and more reliably than anyone else.

The critical question for every engineering leader is: which quadrant are you in, and how do you move to the top right?



Conclusion: Move Fast and Don't Break Things

AI in software engineering is no longer optional. The speed and productivity gains are too significant to ignore. However, as our research shows, realizing the true value of AI requires more than just faster coding. It requires a holistic approach that pairs upstream speed with downstream intelligence.

Organizations that focus only on AI-assisted coding are setting themselves up for failure. They are creating a system that is fast but fragile, where the time saved by developers is paid for in the form of failed deployments, security vulnerabilities, and manual rework.

The path forward is clear. To escape the AI Velocity Paradox and become a true Velocity Leader, engineering organizations must:

1. **Audit the Entire Pipeline**: Identify the automation gaps in testing, security, and deployment processes.
2. **Pinpoint the manual handoffs** and sources of friction.
3. **Shift AI Investment Downstream**: Prioritize the use of AI in the post-coding lifecycle. Implement AI-powered testing, security scanning, and deployment verification to create a safety net for developers.
4. **Consolidate and Standardize**: Move away from a fragmented collection of point tools and toward a unified platform for software delivery. This reduces complexity and allows for the consistent application of AI-powered governance.

“Purpose-built platforms that automate the end-to-end SDLC will be far more valuable than solutions that target just one specific task in future,”

- say 81% of respondents

By embracing AI for everything after coding, organizations can finally unite speed with resilience. They can empower their teams to move fast and not break things, unlocking the full, transformative potential of AI in software development.

Methodology

This report is based on a survey of 900 engineers, platform leaders, and technical managers, commissioned by Harness and conducted by independent research firm Coleman Parkes in August 2025. The sample included 500 respondents in the United States, 200 in the UK, and 100 in each of Germany and France.

THE STATE OF

AI in Software Engineering 2025