THE STATE OF

DevOps Modernization 2026

The AI Multiplier Effect

Triggering the Velocity Paradox by Sacrificing Stability for Speed

harness

Introduction 02

Analysis Framework 03

1. Deployment Risk and Recovery Signals 04

2. Security, Compliance and Performance Strain 06

3. Manual Toil and Downstream Load 08

4. A Fragmentation and Coordination Tax 11

5. Confidence vs Outcomes 13

Conclusion: Maturing your DevOps Functions 15

Methodology 16

harness

Introduction

Since the term DevOps emerged over 15 years ago, toolchains have evolved incrementally. They are assembled under delivery pressure, extended tool by tool, and optimized for speed first, with coherence coming later. However, in many organizations quality gates, verification steps, and incident recovery processes still rely heavily on human coordination and late–stage heroics to keep releases on track.

The surge in AI coding represents a structural acceleration layered on top of these already complex systems. In just a few short years, AI use has become pervasive across the industry. But is the technology mitigating developer burnout and reducing delivery times, or adding to them?

To find out more about the impact AI is having on developer teams, their working practices and their output, we commissioned a survey of 700 engineering practitioners and their managers from large enterprises across the United StatesU.S., United KingdomU.K., France, Germany, and India.

This report examines whether increased AI coding frequency correlates with measurable strain across downstream delivery, stability, security, compliance, and operational resilience indicators.

Analysis Framework

Unless otherwise stated, findings in this report are based on cross-tabulation against:

"How frequently do you, or developers in your team(s), use AI tools for each of the following tasks? Coding"

Segments referenced throughout:

- Very frequently (multiple times per day)
- Frequently (daily)
- Occasionally (weekly)
- Occasionally (monthly or less frequently)

Chapter 1 : Speed Gains, Stability Strain

Deployment Risk and Recovery Signals

Developer teams should be the engine driving corporate innovation and growth. But too often their talent is stifled by commercial pressure to deliver more, at an ever-greater velocity. Over a third (35%) of respondents report daily or even more frequent product deployments, while a similar number (36%) claim they deploy multiple times per week.

Those using AI tools very frequently to help them code are even more likely (45%) to cite daily or faster deployments, versus frequent (32%) and occasional (15%) users. One thing is clear: the days of quarterly or monthly release cycles are long gone. But does speed translate to quality? For many, unfortunately, not.

69%

of very frequent AI coding users say AI-generated code leads to deployment problems at least half the time. Across all respondents, 51% agreed they had these concerns.

For very frequent AI coding tool users, 22% of code deployments result in a rollback, hotfix, or customer-impacting incident. The figure declines for frequent (20%) and occasional (15%) users.

What's more, MTTR for production incidents related to code deployments is likely to be longer for very frequent AI coding tool users (7.6 hours) than for those who use such tools frequently (6 hours) or occasionally (6.3 hours).

Chapter 2 :

Security, Compliance, and Performance Strain

The risks we see are not just outages; they also include security issues, non-compliance with policy, and performance problems.

50%

of very frequent users cite more vulnerabilities and security incidents since their organization started using AI coding tools, versus 32% of frequent users and 26% who occasionally use the tools.

Half (50%) of very frequent users also complain of more non-compliance issues since they started using such tools, versus 31.3% of frequent and 26.3% of rare users.

Some 49% of very frequent users complain of more problems with performance issues (versus 30% and 25% respectively for the other groups).

And over half (51%) of very frequent users (versus 35% and 34%) cite the emergence of more code quality and efficiency problems.

There's no explicit causal link between the use of AI coding and these mounting challenges.

86% of engineering leaders and practitioners argue that security and compliance checks need to be more automated to meet delivery timelines.

Chapter 3 : Automation Imbalance Across the SDLC

Manual Toil and Downstream Load

Most Popular Use Case For AI

Coding is by far the most popular use case for AI, with 84% of respondents claiming they use the technology at least daily for related tasks. However, there's a big gulf between this and the figures for downstream activities like QA testing (68%), performance/cost optimization (63%),and refactoring (62%).

What is clear is that AI-assisted coding isn't the panacea many believe. **In fact, very frequent users are more likely (47%) than frequent (29%) or rare (28%) users to complain that manual downstream work, such as QA, code reviews, and remediation, has become more problematic.** There are clearly teams having more success, as 39% of very frequent users report that manual downstream work is less problematic. Appropriately, occasional users are the most likely to report "no difference" (30%).

of developers' time is spent on repetitive manual tasks such as copying and pasting configuration, obtaining human approvals, chasing tickets, and rerunning failed jobs, according to respondents' estimates. The time is higher for very frequent AI coding users (38%) than for frequent (35%) or rare users (32%).

Seven-in-ten (69%) admit to wasting time due to slow or unreliable CI/CD pipelines, and believe it contributes to developer burnout in their organization “significantly” or “somewhat.”and believe it contributes to developer burnout in their organization “significantly” or “somewhat.” The figure is even higher for very frequent AI coding users (79%) than for those who use these tools frequently (67%) or occasionally (63%).

Pipeline quality is a key concern,

70% of respondents agreeing that their “pipelines are plagued by flaky tests and deployment failures.”

We again saw the percentage agreeing rise in the cohort that used AI coding very frequently (79%).

This drag is taking its toll on work-life balance. **75% say pressure to ship quickly has contributed to burnout.**

71%

of developers are required to work evenings or weekends at least once per week because of release-related tasks or production issues. Almost a third (31%) are called in several times each week.

The share of respondents claiming they have to work evenings and weekends "a few times a month" or more is 96% for very frequent AI coding tool users. This drops slightly for frequent users (95%) and further still for those who occasionally use such tools (66%). In this context, AI coding tools appear to be increasing workload, rather than relieving it.

There's no clear mechanism for AI coding assistants to cause more flaky tests or deployment failures. **We hypothesize, instead, that the teams using these tools most are attempting to deliver more code more frequently and feel the impact of flakiness more acutely as a result. Their pipelines aren't worse, but their needs are greater and they aren't being satisfied.**

Chapter 4 :

A Fragmentation and Coordination Tax

Whether developer tools are AI-powered or not, when practitioners are forced to switch between different solutions for code generation, documentation, testing, and deployment, it can be mentally draining and sap productivity.

78%

complain of fragmented delivery toolchains. The figure is even higher for very frequent users of AI coding tools (83%). It falls slightly for frequent (76%) and rare users (71%).

The opportunities from AI will surely increase as organizations mature their SDLCs with standardized templates and “golden paths” for services and pipelines.

73%

of engineering leaders and practitioners say “hardly any” development teams have these in place.

Code acceleration without golden pathways can negatively impact delivery pipelines. Three-quarters of respondents cite repetitive manual work as a key factor.

More than three-quarters (77%) say teams often need to wait on others for routine delivery work before they can ship code. That rises to 82% for very frequent AI coding tool users (versus 74% and 76% for the two other groups).

Chapter 5:

Confidence vs Outcomes

At the heart of this report is a dichotomy. DevOps teams are under great pressure to deliver at speed and scale. This is driving burnout, and, in the case of AI coding, correlates with downstream issues that could lead to service outages, lost revenue,, and eroded trust. Yet many still rate their DevOps capability highly.

Diving deeper, we found that

72%

of developers and their managers say their "current ways of working will not be sustainable over the long term." The figure surges to 81% of very frequent users of AI coding tools (dropping to 68% and 60% for the other two cohorts).

95%

of very frequent AI coding tool users and 89% of frequent users rate their DevOps capability as good or very good. The figure falls to 65% of those who use AI coding tools only occasionally.

As we saw in Chapter 1, those using the most AI are delivering the fastest. While they also show signs of pain throughout this report and acknowledge the need for further improvement, their current DevOps setups are rated highly, which makes sense. Teams that are deploying to production multiple times a day should take pride in that accomplishment. The challenges evident in this report point to systems at a breaking point.

Are our DevSecOps systems actually good? The high incident rates, slow MTTR, growing compliance issues, and contribution to burnout make this debatable. What's clear is that there's room to improve the quality and coherence, and most teams know some improvement is needed.

Conclusion:

Maturing Your DevOps Function

It's important to remember that AI can deliver fantastic benefits for under-resourced and under-pressure developer teams – especially in coding, but also across the entire SDLC. But tools may also introduce cost, complexity, and quality issues that increase risk and add delays to delivery pipelines. We've documented a clear correlation between AI-assisted coding frequency and deployment issues, rollbacks, MTTR, downstream workload, and more.

The challenge for DevOps teams is clear. Coding assistants are accelerating the capacity of your development teams to innovate. If they can create twice as many changes, your pipelines must improve to cut the risk of making each change by half or better.

If AI can produce four times as many changes, your pipelines must make it four times easier to find a fix for a security issue; four times less likely that a deployment causes an outage; and those outages must be resolved faster, too. And as coding assistants make it trivial to create a new application, stamping out these robust pipelines must be equally easy.

The key is to approach AI and automation not as a replacement for human talent, but a way to optimize your teams, with appropriate oversight and guardrails built in from the start.

Begin by deploying standardized, templatized, and governed pipelines – including feature flags, automated rollbacks and centralized secrets management. With these foundations in place, faster innovation, reduced risk, and greater efficiency could be just around the corner.

Methodology

This report is based on a survey of 700 engineering practitioners and their managers from large enterprises, commissioned by Harness and conducted by independent research firm Coleman Parkes in February 2026. The sample included 300 respondents from the United States, and 100 each in the U.K., Germany, France, and India.