OPSERA

AI Coding Impact 2026 Benchmark Report

Table of Contents

Executive Summary

AI coding assistants are now table stakes for enterprises. However, adoption alone no longer drives advantage. Despite the fact that close to **90% of developers used AI daily in 2025**, outcomes (delivery speed, stability, security, and ROI) varied widely across industries.

Productivity gains are real: Time-to-PR improves 48–58%, and ROI can be achieved in 1–3 months. However, AI also amplifies hidden risks: AI-generated pull requests spend nearly 5 times longer in review, security vulnerabilities rise 15–18%, and junior engineers realize much smaller efficiency gains than seniors, highlighting both talent and governance gaps.

This benchmark report, gathered from more than 250,000 developers in 2025, highlights three major findings:

1. **Governance is the new differentiator.**

   Most organizations have crossed the adoption threshold, but few have built the controls to translate usage into durable business value.

2. **AI accelerates delivery, but amplifies hidden rework.**

   AI speeds up the inner loop, but there's a bottleneck during review: AI-generated PRs wait almost 5 times longer for review than manual code.

3. **There's a lot we still don't know.**

   Many leaders can answer "Do we have licenses?" but cannot answer other critical questions:

   » Where is AI improving throughput versus simply increasing code churn?

   » Which teams are benefiting, and which are taking on hidden risk?

   » Are we trading short-term speed for long-term stability, compliance, or cost?

Key Benchmarks

48–58%
Average Time-to-PR improvement

AI-generated PRs wait
4.6x longer
for review

AI code shows
15-18%
more security vulnerabilities

ROI payback period
1–3 months
when usage is actively managed

About this report

Before we take a look at industry trends and outcomes, it is important to know what this data is and where it comes from. The insights in this report are drawn from a single, consistent set of **data collected through the end of Q4 2025** by Opsera's product team. Data also comes from GitHub Copilot, Cursor, Claude Code and other tools.

Copilot CURSOR Claude

Data Sources and Scope

Data sourced from

250,000+
developers

60+
enterprise organizations

Q1–Q4 2025
verified data

Industries included

- Technology
- Startups
- Banking
- Healthcare
- Insurance
- Manufacturing

The data in this report goes beyond developer sentiment surveys, focusing on actual behaviors in production environments. Our dataset combines usage-level telemetry with **outcome-based metrics across the software delivery lifecycle.**

How to use this report

This report is designed to serve as a strategic guide, using industry-relative comparisons to provide context on the current state of AI adoption and performance. The primary value of this document lies in trend identification, rather than serving as a tool for point-in-time optimization or tactical adjustments. Users should leverage these insights for strategic planning and understanding evolutionary changes within the AI landscape.

AI Adoption vs. AI Acceptance Benchmarks

At a weighted average of **90%, AI adoption is flatlining**. Most industries have already adopted AI as part of their development tool stack.

Acceptance rates tell a more detailed story in each industry:

- Acceptance rates that are **too low** signal distrust of AI tools, poor prompts, or weak workflows
- Acceptance rates that are **too high** indicate rubber stamping and higher risk

Ideally, acceptance rates should sit somewhere in the middle.

However, acceptance rates alone should not be seen as a predictor of success.

Out of context, acceptance rates can be misleading. High acceptance rates do not guarantee that AI-generated code survives review, avoids rollback, or remains in production two weeks later.

Adoption vs. Acceptance by Industry

Industry	Adoption Rate	Acceptance Rate	What This Signals
Startups	97%	~46–51%	High trust, fast iteration, higher risk tolerance
Technology / SaaS	95%	~41–46%	Mature usage with selective filtering
Banking / Finance	89%	~31–36%	Disciplined review, compliance-driven caution
Healthcare	87%	~29–34%	High validation burden, safety-first governance
Insurance	84%	~31–36%	Manual controls slow trust expansion
Manufacturing	85%	~33–38%	Moderate trust, workflow inconsistency
Other Industries	89%	~34–39%	Mixed maturity and governance models
Weighted Enterprise Average	91%	~36–41%	Adoption saturated, trust uneven

Adoption = who has AI available
Acceptance = how often AI suggestions are kept in final code

Productivity Impact Benchmarks

AI reliably speeds up the inner loop for every industry in this report, dramatically **reducing active coding time** for teams across the board.

Elite teams average **under 54 minutes** of active coding per PR.

However, that coding speed isn't translating to speed later in the development process. Testing and security slows down development, particularly in regulated industries like healthcare and finance.

Productivity Impact Benchmarks by Industry

Industry	Time-to-PR	Deployment Frequency	Code Velocity	Features/Sprint
Startups	69–74% faster	+52–62%	+62–72%	+47–57%
Technology / SaaS	64–68% faster	+57–67%	+57–67%	+42–52%
Banking / Finance	44–54% faster	+32–42%	+37–47%	+27–37%
Healthcare	34–44% faster	+46–56%	+32–42%	+22–32%
Manufacturing	39–49% faster	+47–57%	+34–44%	+24–34%
Insurance	29–39% faster	+42–52%	+27–37%	+20–30%
Other Industries	42–52% faster	+47–65%	+40–50%	+30–40%
Weighted Enterprise Average	50–60% faster	+39–49%	+44–54%	+34–44%

Percentage improvement vs. pre-AI baseline

The Wait Time Anomaly

AI-assisted development dramatically accelerates coding, but the gains stall at review. While Time-to-PR improves by **48–58% on average**, AI-generated pull requests wait **4.6× longer** to be picked up for review than human-written PRs.

This wait time spike occurs primarily before first review, not during testing or deployment, and persists across industries.

The result is a throughput illusion: inner-loop speed increases, but outer-loop friction neutralizes much of the benefit. Without automated review, testing, and trust-building controls, AI shifts work downstream rather than eliminating it, turning velocity gains into queue time.

Where Wait Time Occurs in the SDLC

SDLC Phase	Observed Impact
Active coding (inner loop)	Accelerated by 48–58% (Time-to-PR)
PR submission → first review	Primary wait time bottleneck
Review → merge	Extended due to higher cognitive load
Post-merge validation	Increased rework when AI-generated issues surface

The wait time spike is concentrated **before review begins**, not during execution or deployment.

Quality Outcome Benchmarks

If leaders are just looking at surface-level quality metrics, AI-assisted development can create a false sense of security.

While AI-generated code shows slightly higher test pass rates and lower bug escape rates, it also introduces **more security vulnerabilities** and consistently higher **code duplication**.

These risks are largely invisible during initial delivery because AI is very good at producing code that looks correct and passes tests. However, without scaled security scanning, refactoring discipline, and governance, organizations trade short-term velocity for long-term exposure.

In other words, AI does not reduce risk: it **front-loads speed while back-loading failure** unless executive oversight shifts from adoption metrics to outcome and control metrics.

AI-Assisted vs. Manual Code

Metric	AI-Assisted Code	Manual Code
Test Pass Rate	93%	92%
Bug Escape Rate (per 1K LOC)	2.8	3.1
Code Duplication	13.5%	10.5%

Cross-industry comparison

Quality Outcomes by Industry

Industry	Test Pass Rate	Bug Escape Rate (per 1K LOC)	Code Duplication
Technology / SaaS	93% vs 91%	2.8 vs 3.2	14% vs 11%
Banking / Finance	95% vs 94%	2.1 vs 2.3	12% vs 9%
Healthcare	96% vs 95%	1.8 vs 1.9	11% vs 8%
Insurance	94% vs 93%	2.4 vs 2.6	13% vs 10%
Manufacturing	91% vs 90%	3.2 vs 3.5	15% vs 12%
Startups	90% vs 88%	4.2 vs 4.8	16% vs 13%
Other Industries	92% vs 91%	3.0 vs 3.3	13% vs 10%
Weighted Enterprise Average	93% vs 92%	2.8 vs 3.1	13.5% vs 10.5%

AI-Assisted vs. Manual

Security Risk Benchmarks

While AI code has improved many aspects of the development process, it's not perfect.

AI-generated code contains 15–18% more security vulnerabilities than human-written code across all industries.

The highest exposure is found in regulated environments like healthcare and insurance.

While AI improves visible quality signals, many introduced flaws are subtle, logic-based, and integration-driven, allowing them to pass tests and evade manual review. At the same time, increased code volume overwhelms traditional security processes, pushing risk downstream where remediation is more costly. Without automated scanning and policy enforcement embedded in CI/CD, AI accelerates delivery while quietly increasing security exposure.

In other words, AI does not reduce security work; it **front-loads** it.

Vulnerabilities per 10K Lines of Code (AI vs. Manual)

Industry	AI-Assisted Code	Manual Code
Technology / SaaS	3.5	2.9
Banking / Finance	4.2	3.8
Healthcare	6.5	5.2
Insurance	5.8	4.9
Manufacturing	4.8	4.1
Startups	4.5	3.7
Other Industries	5.2	4.5
Weighted Enterprise Average	4.6	3.9

License Utilization and ROI Benchmarks

You might buy the licenses, but will devs actually use them? The answer seems to be "mostly," with an average usage of 79% across all industries in this report. However, this means

21% of licenses aren't being used, and that impacts ROI.

Under-utilization of licenses is a silent tax on ROI. The more licenses are used (and the more deeply they're used) the faster AI tools pay for themselves.

Teams using AI for design, refactoring, and test generation save **~3 workdays per ticket,** However, teams that use AI for autocomplete only will see marginal returns on investment.

License Utilization and ROI Benchmarks by Industry

Industry	License Utilization	Annual Cost / Dev	Productivity Gain	ROI Payback Period	Annual Savings / Dev
Technology / SaaS	~90%	$228–$240	55–65%	1–2 months	$11K–$13K
Startups	~85%	$204–$228	60–70%	1–2 months	$10K–$12K
Banking / Finance	~80%	$228–$240	35–45%	2–3 months	$7K–$9K
Healthcare	~72%	$228–$240	30–40%	3–4 months	$6.5K–$8.5K
Insurance	~70%	$228–$240	35–45%	3–5 months	$5K–$7K
Manufacturing	~60%	$228–$240	45–65%	3–4 months	$6.5K–$8.5K
Other Industries	~75%	$228–$240	38–48%	2–3 months	$7.5K–$9.5K
Weighted Enterprise Average	~79%	$228–$240	42–52%	2–3 months	$8K–$10K

AI Coding Assistant Tools

Over the past several months, the AI coding assistant landscape has stratified into clear tiers.

GitHub Copilot remains dominant with roughly **60–65% market share and 30–40% acceptance rates.** Cursor has emerged as a fast-growing challenger while Claude Code represents a newer, agentic class, showing the **highest acceptance rates (38–48%)** as it handles multi-file, task-level changes. A long tail of specialized tools adds incremental value but fragments visibility.

Tools materially shape workflow, risk profile, and governance burden; autocomplete tools optimize inner-loop speed, while agentic tools reshape workflows and risk profiles, making unified measurement and SDLC-wide governance essential as AI adoption scales.

Tool / Category	Estimated Market Share	User Scale	Avg. Acceptance Rate	Primary Workflow Impact	Governance Implications
Copilot	~60–65%	~20M users, ~1.3M paid	30–40%	Inline autocomplete, fastest Time-to-PR gains	Relies on existing review & testing rigor; low visibility into downstream outcomes
CURSOR	~10–12%	1M+ DAU	35–45% (est.)	IDE-centric, repo-aware workflows, faster onboarding	Higher leverage per developer; requires tighter review discipline
Claude	~12–15%	Rapid enterprise growth	38–48% (est.)	Agentic, multi-file & task-level changes	Largest blast radius; demands strong guardrails and policy enforcement
Other Tools	~2–8%	5–8M combined	28–38%	Specialized tasks (review, refactoring, testing)	Fragmented visibility; increases governance complexity
Autocomplete (Category)	Majority share	Broad adoption	Stabilizing	Accelerates inner-loop coding speed	Speed outpaces review without automation
Agentic Tools (Category)	Fast-growing	Early-stage scale	Rising fastest	Changes unit of work, not just speed	Requires SDLC-wide controls and monitoring
Agentic DevOps (Category)	Fast-growing	Early-stage scale	Rising fastest	Changes unit of work, not just speed	Requires Enterprise wide controls and monitoring

What does the data reveal as a whole?

1. Adoption Has Peaked, Outcomes Have Not

AI coding assistant adoption has converged across industries, reaching a **~90% enterprise average**. At this level, adoption no longer predicts performance. Organizations with identical adoption rates show dramatically different outcomes in productivity, quality, security, and ROI, indicating that how AI is governed matters more than whether it is present.

How **AI is governed** matters more than whether it is present.

AI-generated PRs wait **4.6× longer**

2. Inner-Loop Speed Outpaces Outer-Loop Capacity

AI consistently reduces Time-to-PR by 48–58%, but these gains stall at review. AI-generated PRs wait 4.6× longer for review, making wait time the dominant bottleneck. When review, testing, and security workflows remain manual, AI shifts work downstream instead of increasing throughput.

3. Productivity Gains Are Real but Uneven

All industries experience meaningful productivity improvements, but the magnitude varies widely. Technology and startups convert AI speed into deployment velocity, while regulated industries see diminishing returns as compliance and validation gates absorb the gains. AI amplifies existing delivery systems rather than equalizing them.

AI **amplifies existing delivery systems** rather than equalizing them.

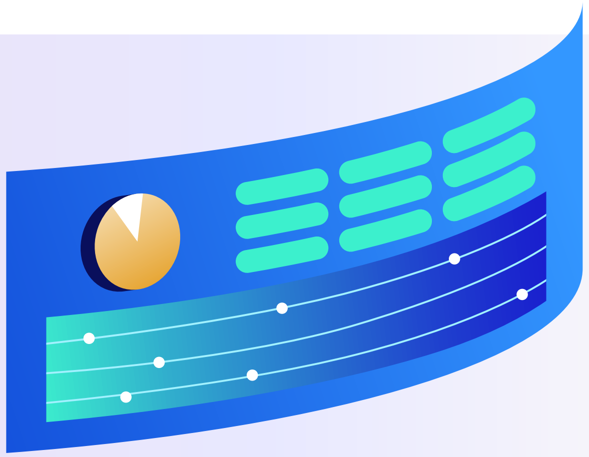
AI-assisted code provides modest quality improvements but also produces **13.5% vs. 10.5%** higher code duplication and **15–18%** more security vulnerabilities

4. Quality Improves Slightly While Risk Increases

AI-assisted code shows modest quality improvements, including higher test pass rates (93% vs. 92%) and lower bug escape rates (2.8 vs. 3.1 per 1K LOC). However, these gains are paired with higher code duplication (13.5% vs. 10.5%) and 15–18% more security vulnerabilities, creating long-term technical and security debt if unmanaged.

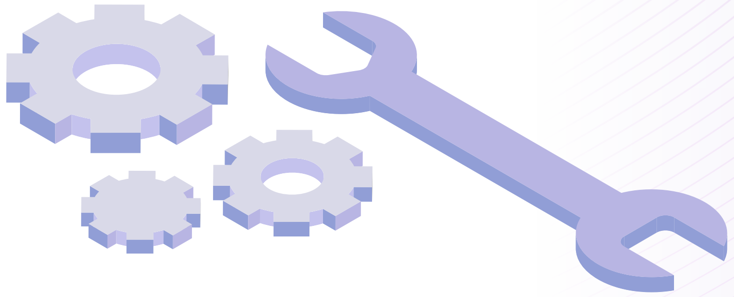
5. ROI Is Fast but Fragile

AI tools typically deliver ROI within 1–3 months, but realized value depends on license utilization and workflow maturity. Underutilized licenses and unmeasured usage dilute returns, while hidden rework and review delays erode productivity gains over time.



6. Tool Choice Shapes Risk Surface

Autocomplete tools primarily accelerate code creation, while agentic tools alter the unit of work itself. As organizations adopt multiple tools in parallel, **fragmented visibility becomes a risk**, making unified measurement essential to avoid blind spots.



7. Measurement Gaps Are the Largest Barrier to Sustainable Value

The strongest predictor of long-term success is not adoption or acceptance, but **end-to-end visibility.** Organizations that track retention, wait time, change failure, and security outcomes convert AI speed into durable value. Those that rely on vanity metrics generate more code without knowing whether it helps or harms the business.

Executive Implications

AI-assisted development is no longer a tooling decision. It is an operating model decision. Leaders who treat it as an add-on will see short-term gains and long-term drag. Leaders who act now can convert AI speed into durable advantage.

What Leaders Should Act On Now

1. **Shift accountability from adoption to outcomes**

   With adoption near saturation, tracking licenses or usage **no longer predicts performance**.

   Mandate reporting on outcome metrics executives actually care about: review wait time, suggestion retention, change failure rate, vulnerability density, and realized ROI. If these are not visible, velocity metrics should not be trusted.

2. **Collapse review and validation bottlenecks**

   AI cuts coding time by ~50%, but AI-generated **PRs wait 4.6× longer** for review, neutralizing gains.

   Invest in automation before expanding AI usage. Prioritize CI/CD checks, security scanning, and policy enforcement that run faster than humans. Speed that cannot clear review is wasted capacity.

3. **Treat security as a scaling constraint, not a checkpoint**

   AI-generated code carries **15–18% higher vulnerability** density, and risk accumulates quietly as volume increases.

   Require automated security controls as default gates for AI-generated code. Do not rely on manual review to absorb AI risk. If security tooling cannot scale, slow AI expansion.

4. Reframe ROI as capital efficiency, not license payback

AI pays for itself in 1–3 months, but **underutilization** and rework erode value over time.

Tie AI investment to capitalized engineering outcomes. Track utilization depth, retained code, and feature throughput. Reclaim unused licenses and coach teams on higher-leverage use cases.

5. Protect reviewers and institutional knowledge

Reviewers are becoming the safety net for AI output, increasing **burnout and risk**.

Reduce reviewer cognitive load through smaller PRs, automated checks, and clear labeling of AI-generated changes. Human judgment should validate, not decipher.

6. Align speed targets with industry risk tolerance

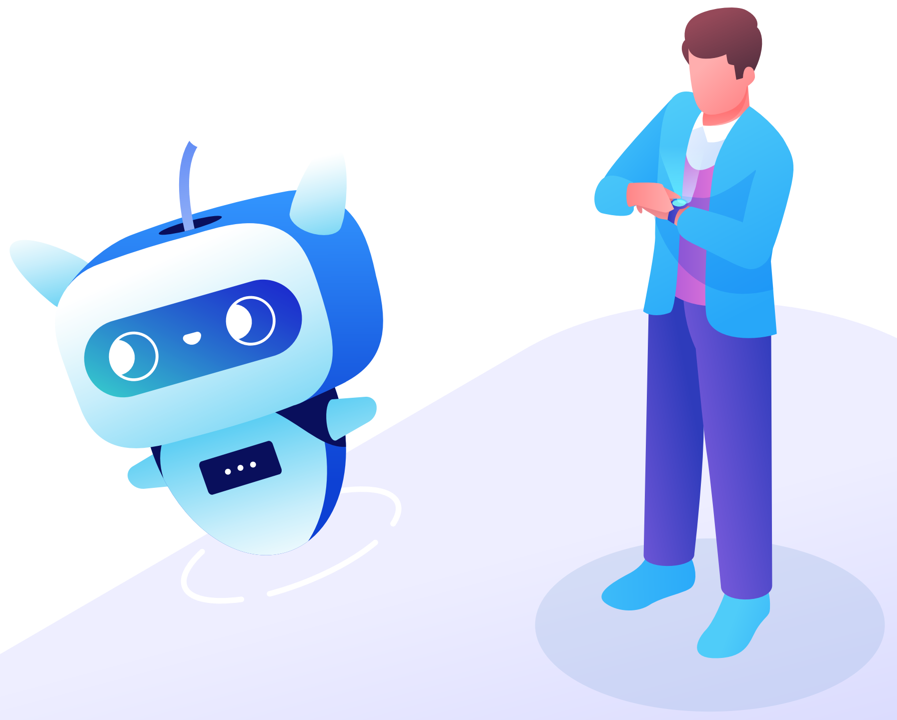
The same AI velocity is safe in a startup and dangerous in **healthcare or finance**.

Define success relative to regulatory and operational risk, not headline productivity benchmarks. Use industry-aware targets for speed, quality, and security.

7. Prepare for agentic AI now, not later

Tools are moving from assistance to autonomy, increasing both l**everage and risk.**

Establish governance frameworks, auditability, and visibility before agentic workflows become widespread. Waiting will compound blind spots.



AI-assisted development has crossed the point of experimentation and entered the realm of enterprise operating risk.

The benchmarks show that while AI reliably accelerates coding and delivers fast ROI, those gains are fragile without governance. Organizations that focus on adoption and velocity alone generate more code, but not necessarily more value, as review delays, security exposure, and hidden rework absorb the speed. **The leaders that outperform are those that treat AI as a managed system**, measuring retention over acceptance, collapsing wait time before expanding usage, and scaling security ahead of generation.

As AI tools move toward greater autonomy, the gap between speed and control will widen. The enterprises that win in 2026 will not be those that generate the most code, but those that convert AI velocity into durable, trusted outcomes.

About Opsera

opsera.io

Opsera is the unified DevOps platform that automates and secures software delivery across clouds, tools, and teams. Trusted by leading enterprises, Opsera turns DevOps complexity into competitive advantage with smart orchestration.