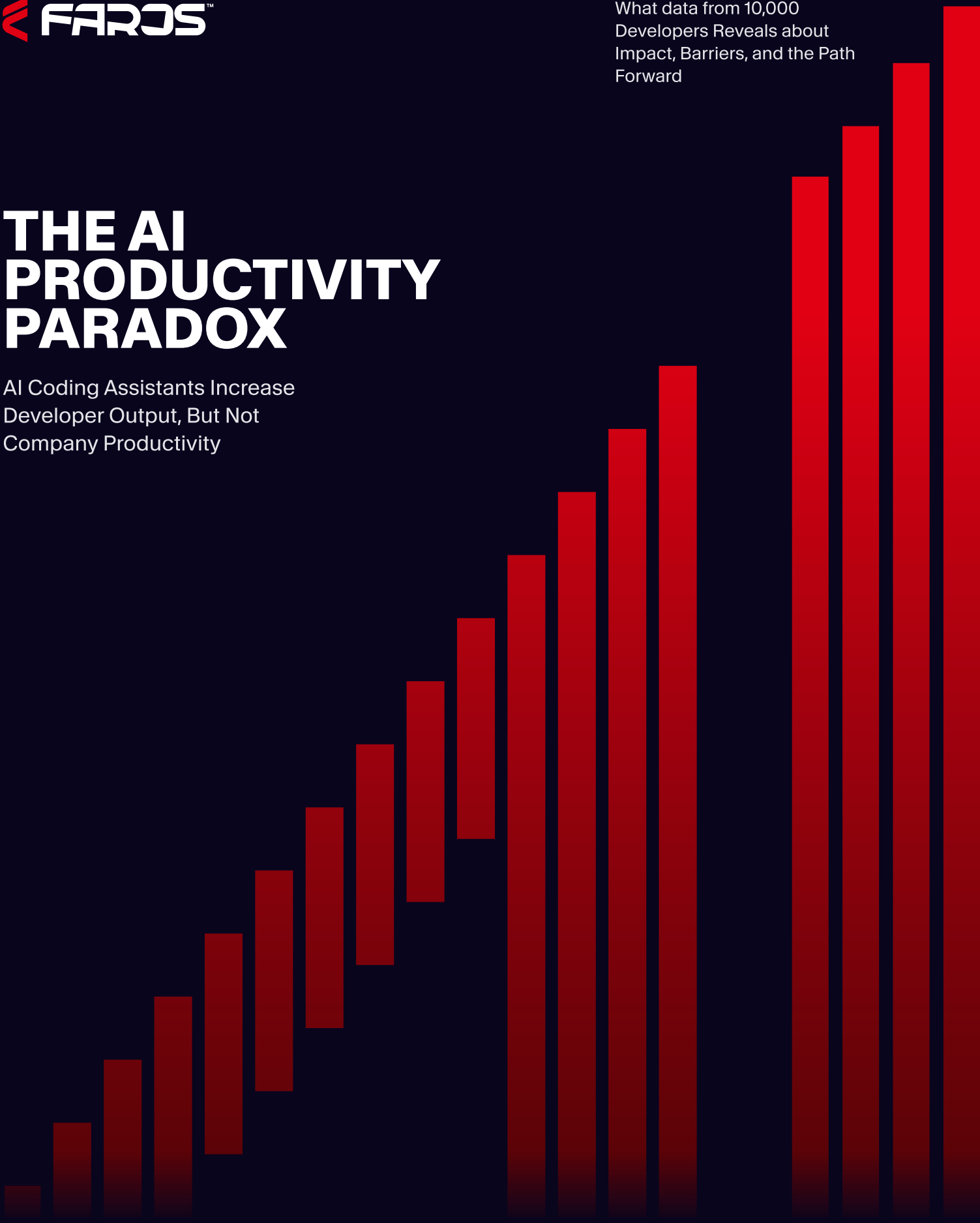
FARROS™

What data from 10,000 Developers Reveals about Impact, Barriers, and the Path Forward

THE AI PRODUCTIVITY PARADOX

AI Coding Assistants Increase Developer Output, But Not Company Productivity



Insights from Faros AI's analysis of 1,255 teams across leading software organizations

JULY 2025

Executive Summary: The AI Productivity Paradox

Faros AI analyzed telemetry from 1,255 teams and over 10,000 developers. The results highlight a consistent pattern: **AI adoption reshapes team behavior, but those changes often stall before reaching the organizational level.**

While AI coding assistants are widespread and boost individual productivity, most companies have not yet seen those gains translate into measurable business impact.

Quantified Team-Level Findings

- **Delivery metrics remain unchanged:** While AI may be changing developer workflows and output, it has **not moved the needle on key delivery metrics** like lead time and change failure rate.
- **Higher throughput, slower reviews:** On average, developers using AI complete **21% more tasks** and **merge 98% more pull requests (PRs)**. At the same time, **PR review time increased by 91%**, indicating that human review remains a bottleneck.
- **More parallel workstreams:** Developers on teams with high AI adoption interact with **9% more tasks and 47% more pull requests per day**. These are significant effects, particularly the increase in task contexts ($\rho = +0.22$).
- **Mixed quality signals:** While we observe a modest correlation between AI usage and positive quality indicators (fewer code smells and higher test coverage), AI adoption is consistently associated with a **9% increase in bugs per developer and a 154% increase in average PR size**.

No Measurable Organizational Impact

When viewed at the company level, these team-level gains do not scale. **Correlations between AI adoption and organization-wide delivery metrics are weak or nonexistent.** This is due to uneven adoption, workflow bottlenecks, fragmented tooling, and the lack of coordinated enablement strategies.

Achieving Business Value and AI ROI at Scale

Qualitative fieldwork and operational insights from companies that are seeing AI performance gains reveal the foundational investments required in **enablement, platforms, and strategy** to provide a measurable return on investment:

- **Strategy:** Define clear goals, usage guidelines, and change management strategies
- **Training:** Provide guided onboarding, ramp-up, and role-specific training programs
- **Playbooks:** Build internal playbooks and communities of practice to nurture wins
- **Platform Engineering:** Double down on platform foundations to support rapid AI experimentation and faster flow of code through development pipelines
- **Visibility:** Implement AI engineering consoles to create a centralized data-driven command center for monitoring effectiveness and safety

To scale AI impact and prepare for the **upcoming wave of agentic development** (where autonomous systems participate in code planning, generation, and review alongside human developers), this report outlines **five key enablers: Workflow design, governance, infrastructure, training, and cross-functional alignment.**

Table of Contents

State of the Industry

A New Era Begins—But Foundations Still Matter

AI adoption in software engineering has outpaced almost every recent technology shift. Over 75% of developers now use AI coding assistants weekly, and expectations for productivity gains are surging.

But many organizations report a disconnect. While developers say they're working faster, companies aren't seeing measurable improvement in delivery velocity or business outcomes. A 2024 McKinsey survey found that eight in ten companies report no material genAI impact on earnings.

This report investigates that gap. Drawing on telemetry from over 10,000 developers, it surfaces where AI is already changing developer behavior—and why those changes haven't yet scaled. For leaders looking to unlock AI's full potential, the data points to both promising leverage and persistent friction.

Pulse Check: Are Productivity Gains Materializing?

Despite rapid AI adoption and widespread optimism, the impact on engineering productivity remains unclear.

Studies at the individual level report measurable benefits: a 2024–2025 study by MIT, Princeton, and Microsoft found that AI-assisted developers completed 26% more tasks and compiled 38% more frequently. But organizational data tells a different story. Most companies see little to no change in overall delivery performance.

Faros AI's analysis confirms this pattern. While team-level changes are measurable, company-wide delivery metrics—including throughput, lead time, and incident resolution—remain flat. These results echo the 2024 DORA report, which similarly found that modest individual gains often vanish amid unaddressed delivery bottlenecks.

Coding may be faster. Delivery is not—yet.

Part I: Achieving Measurable Productivity Gains with AI

From Promise to Reality: Where AI Productivity Stalls

AI has entered the software development mainstream with unprecedented speed. As more and more developers integrate AI into their workflows, senior leaders increasingly expect these technologies to boost productivity, speed delivery, and improve quality. But a key question remains: Are those expectations being met?

Early developer-level data suggests yes, with time savings, cognitive relief, and faster code writing widely reported. But there is also evidence that the existing systems around them aren't moving fast enough to keep up.

The following section presents empirical findings from Faros AI's analysis of over 10,000 developers across 1,255 teams. The results reveal where AI is already making a difference and where performance is stalling due to infrastructure constraints, uneven adoption, and a lack of strategic alignment.

As the industry enters the age of agentic development, these gaps must be closed quickly. The organizations that move now will build a durable advantage. Those who wait will find that AI alone cannot outrun a broken system.

Scope and Methodology

This study analyzes the impact of AI coding assistants on software engineering teams, based on telemetry from task management systems, IDEs, static code analysis tools, CI/CD pipelines, version control systems, incident management systems, and metadata from HR systems, from 1,255 teams and over 10,000 developers across multiple companies. The analysis focuses on development teams and covers up to two years of history, aggregated by quarter, as teams increased AI adoption.

We define AI adoption in this report as the usage of developer-facing AI coding assistants—tools including GitHub Copilot, Cursor, Claude Code, Windsurf, and similar. These are generative AI development assistants that integrate directly into the software development workflow—typically through IDEs or chat interfaces—to help developers write, refactor, and understand code faster. Increasingly, these tools are expanding beyond autocomplete to offer agentic modes, where they can autonomously draft pull requests, run tests, fix bugs, and perform multi-step tasks with minimal human intervention.

To isolate the relationship between AI adoption and engineering outcomes, we:

- Standardized all metrics per company to remove inter-org variance
- Used Spearman rank correlation (ρ) to assess relationships of metrics to AI usage
- Reported only those metrics with data from ≥6 companies and statistically significant correlations (p-value < 0.05)
- For each team, we calculated the percent change in metric values between the two quarters with the lowest AI adoption and the two quarters with the highest
- Excluded outlier data and metrics with insufficient historical coverage

This approach enables comparisons within each company over time and avoids misleading aggregate assumptions across different org structures.

Our analysis provides the most statistically grounded view yet of AI's impact on engineering at the team level. The findings reveal clear, measurable changes in how teams operate, although these improvements have not yet translated into meaningful company-wide performance gains.

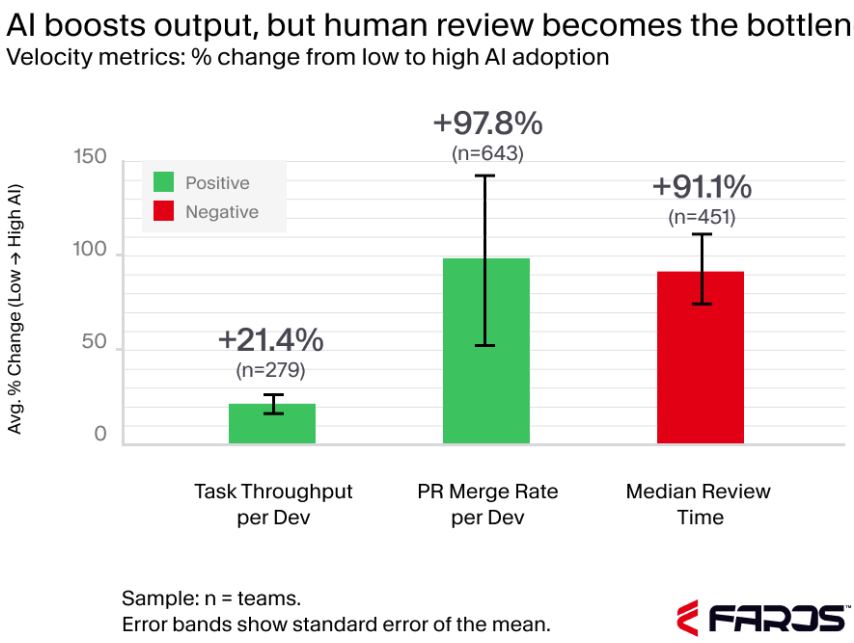
Versioning note: This version of the report reflects analysis as of June 2025. Future editions may expand coverage as AI usage matures across more organizations and product features evolve.

Key Findings at the Team Level

1. Task Throughput Increases, But Review Time Slows

Developer velocity rises as AI usage increases.

On average, developers using AI complete 21% more tasks ($\rho = 0.21$, $p < 0.01$) and merge 98% more pull requests ($\rho = 0.3$, $p < 0.01$). At the same time, PR review time increased by approximately 91% ($\rho = 0.08$, $p < 0.01$), indicating that human review remains a bottleneck.

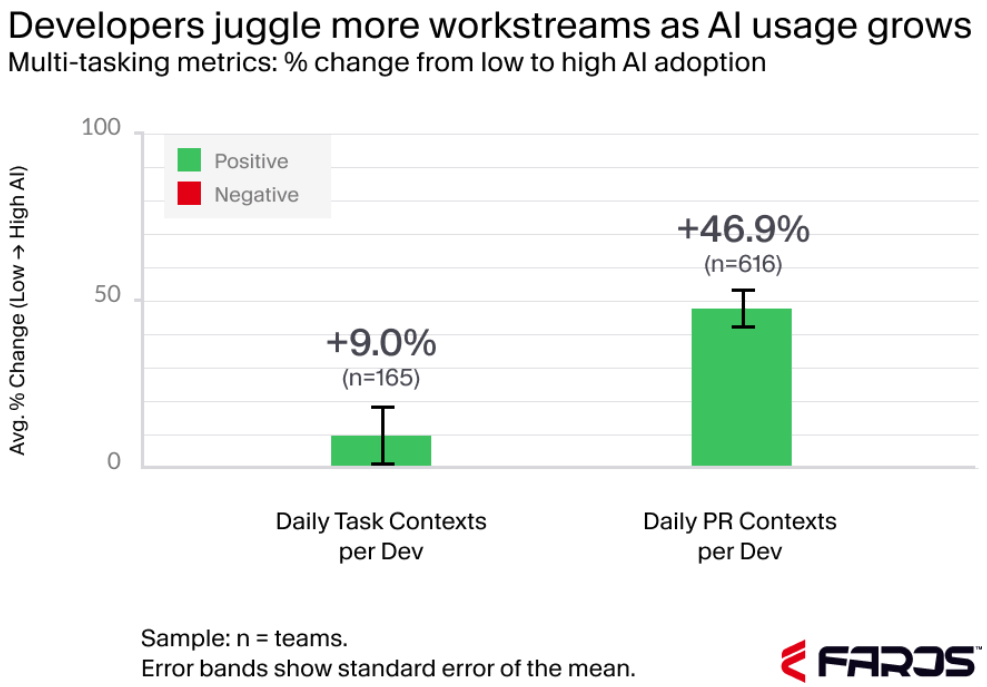


2. Developers Handle More Workstreams Concurrently

Developers on teams with high AI adoption touch more tasks and pull requests per day.

Developers on teams with high AI adoption interact with 9% more tasks ($\rho = 0.22$, $p < 0.01$) and 47% more pull requests per day ($\rho = 0.15$, $p < 0.01$). This likely reflects that developers are completing tasks and pull requests more quickly and can parallelize work more effectively because AI agents are contributing to the workload. With AI coding assistants in use, an engineer can make progress on one task while their agent simultaneously handles another.

Historically, context switching has been viewed as a negative indicator, correlated with cognitive overload and reduced focus. That benchmark is shifting. In the AI-augmented environment, developers are not just writing code—they are initiating, unblocking, and validating AI-generated contributions across multiple workstreams. As the developer's role evolves to include more orchestration and oversight, higher context switching is expected.



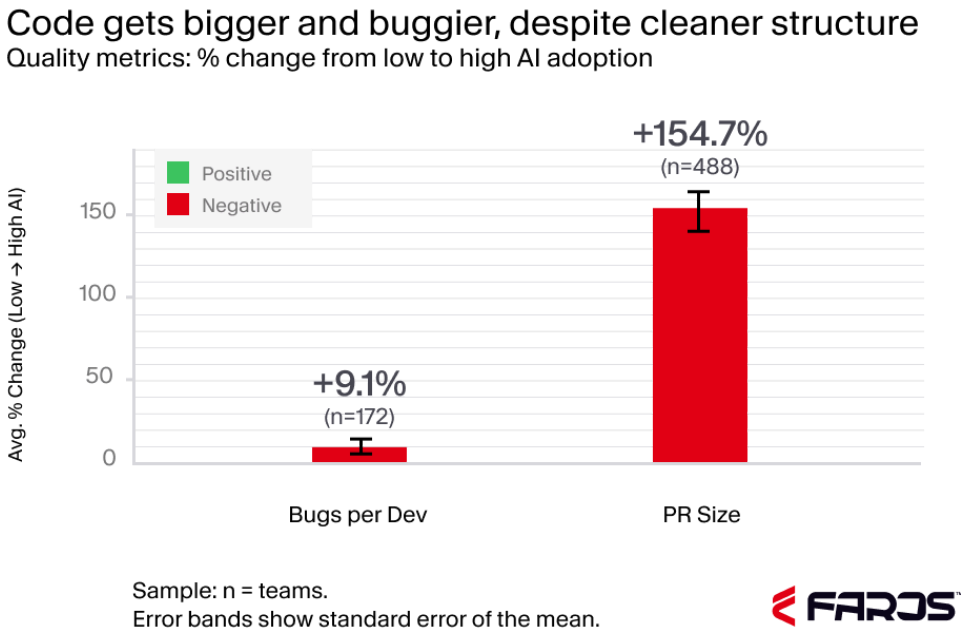
3. Mixed Impact on Code Quality

AI adoption is associated with some improvements in code structure but introduces new risks.

While we observe a modest correlation between AI usage and positive quality indicators, specifically, a reduction in code smells ($\rho = -0.11$, $p < 0.01$) and an increase in test coverage ($\rho = +0.07$, $p < 0.05$), these signals are based on limited time series data and should be interpreted directionally.

By contrast, high AI adoption is consistently associated with a 154% increase in average pull request size ($\rho = 0.07$, $p < 0.01$) and a 9% increase in bugs per developer ($\rho = 0.11$, $p < 0.01$), suggesting that AI-generated code may be more verbose and less incremental, making it harder to review and error-prone.

In short, while AI may support better structure or test coverage in some cases, it also amplifies volume and complexity, placing greater pressure on review and testing systems downstream.



4. Team Software Delivery Performance Remains Mostly Flat

AI adoption shows a limited impact on software delivery as measured by the four DORA metrics.

- Deployment Frequency (how often code is released to production) has improved slightly but significantly among teams with high AI usage ($\rho = 0.09$, $p < 0.02$).
- Lead Time for Change (the time it takes for a pull request to be deployed to production) has gotten longer, driven in part by longer review cycles ($\rho = 0.12$, $p < 0.04$).
- Change Failure Rate (the percentage of deployments causing incidents) and Mean Time to Resolve (MTTR) (the time to resolve production issues) remain flat.

These signals are based on limited time series data and should be interpreted directionally. However, taken together, they suggest that while AI may be changing developer workflows and output, it has not moved the needle on key delivery metrics.

No Measurable Organizational Impact

Despite these team-level changes, we observed no significant correlation between AI adoption and improvements at the company level.

Across overall throughput, DORA metrics, and quality KPIs, the gains seen in team behavior do not scale when aggregated.

This suggests that downstream bottlenecks may be absorbing the value created by AI tools, and that inconsistent AI adoption patterns throughout the organization—where teams often rely on each other—are erasing team-level gains.

To understand why measurable gains at the team level don't scale to the organization, Part II examines adoption patterns that may explain why, systemic barriers blocking gains, and actionable insights into what high-performing companies are doing differently.

Part II: Why Gains Stall and the Structural Barriers to Scaling AI Impact

Observed Adoption Patterns (Limited Dataset)

Based on directional signals from a subset of companies with complete telemetry coverage, we identified **four adoption patterns** that help explain why team-level AI gains often fail to scale.

1. **AI adoption only recently reached critical mass.** In most companies, widespread usage (>60% weekly active users) only began in the last two to three quarters, suggesting that adoption maturity and supporting systems are still developing. Even where overall adoption appears strong, usage remains uneven across teams. And because software delivery is inherently cross-functional, accelerating one team in isolation rarely translates to meaningful gains at the organizational level.
2. **Tool usage is fragmented.** As AI adoption has grown, many organizations now support multiple tools, ranging from GitHub Copilot to Cursor, Claude, Windsurf, and others. This tool sprawl introduces fragmentation and increases the burden on enablement, training, and integration teams.
3. **Adoption skews toward less tenured engineers.** Usage is highest among engineers who are newer to the company (not to be confused with junior engineers who are new to the profession). This likely reflects how newer hires lean on AI tools to navigate unfamiliar codebases and accelerate early contributions. In contrast, lower adoption among senior engineers may signal skepticism about AI's ability to support more complex tasks that depend on deep system knowledge and organizational context.
4. **Most usage remains surface-level.** Across the dataset, most developers use only autocomplete features. Advanced capabilities—such as chat, context-aware review, or agentic task execution—remain largely untapped.

Taken together, these patterns reveal an ecosystem still in early stages, where fragmented adoption, shallow usage, and uneven enablement prevent localized AI gains from translating into system-wide impact.

Systemic Barriers (Based on Client Conversations)

In parallel, qualitative input from CTOs and engineering leaders at Fortune 1000 companies reveals several recurring obstacles to translating team-level gains into broader organizational performance.

1. **Downstream bottlenecks cancel out gains.** AI accelerates code generation, but review, testing, and deployment stages have not kept pace. Increased PR volume overloads reviewers, test frameworks remain brittle, and release pipelines are rarely optimized for higher velocity. According to Amdahl's Law, the system's speed is constrained by its slowest link—without lifecycle-wide modernization, AI's benefits are quickly neutralized.

2. **Grassroots adoption lacks structure and scale.** In most organizations, AI usage is still driven by bottom-up experimentation. While early enthusiasm is high, the lack of centralized enablement creates four common pitfalls:
    - Developers spend time learning tools with little guidance, eroding early time savings.
    - Most users receive no formal training, leading to inconsistent outcomes.
    - No tailored strategy exists by role or tenure, resulting in uneven adoption.
    - Best practices are rarely shared—there are no playbooks, no internal communities.

    The result: limited leverage of advanced AI features and low return on tooling investment.
3. **Strategic direction from engineering leadership has been insufficient.** Many leaders assume that deploying AI tools is enough. In practice, the absence of clear goals, usage guidelines, and change management leads to shallow, inconsistent adoption. Without a top-down strategy aligned to business priorities, AI becomes a disconnected tool rather than a catalyst for transformation.

These barriers explain why AI-driven gains, while measurable at the team level, often dissipate at the organizational level. Without stronger coordination, enablement, and investment in system-wide readiness, gains will remain isolated.

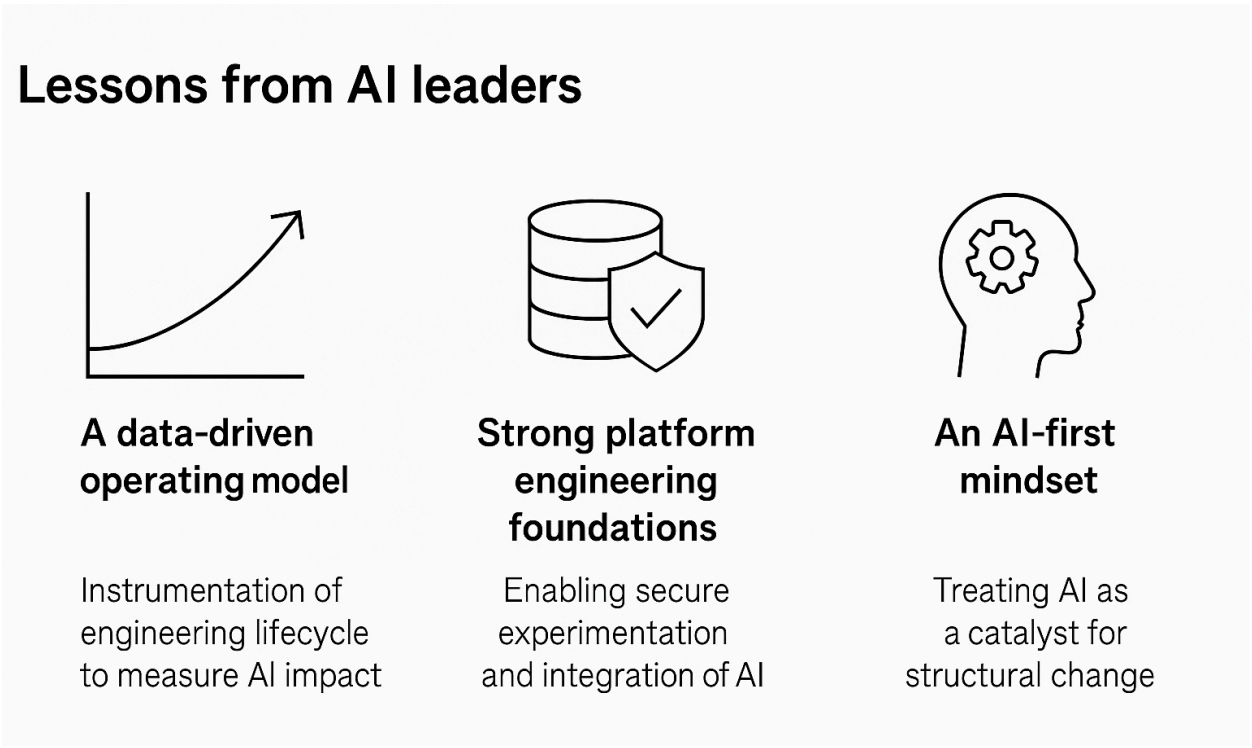
What High-Performing Companies Do Differently

Note: This section is not derived from correlation analysis, but from qualitative fieldwork and operational insight. It outlines the foundational investments and practices that distinguish organizations realizing material gains from AI adoption.

Faros AI's data confirms that incremental AI adoption is yielding minimal results. The organizations making meaningful AI gains are those investing in platform modernization, agent readiness, and comprehensive change management.

Furthermore, with the emergence of agentic systems—AI tools capable of reasoning, planning, and executing tasks across the SDLC **autonomously**—the urgency is growing. Within the next 12 months, companies must shift from experimentation to operationalization or risk falling permanently behind.

In contrast to organizations where AI adoption remains shallow or fragmented, high-performing companies are already capturing measurable benefits. These organizations share three operational characteristics that create the foundation for consistent AI leverage and higher returns on investment.



1. **Data-driven decision making:** They instrument the full development lifecycle, tracking throughput, flow efficiency, test coverage, and code quality to identify bottlenecks and opportunities for AI leverage. This observability both establishes a pre-AI baseline and reveals where AI accelerates value or stalls it.
2. **Strong platform foundations:** They treat AI enablement as a product, with platform teams building centralized prompt libraries, managing model deployment, and supporting telemetry integration.
3. **AI-first mindset:** Top performers treat AI as a catalyst for structural change. These companies explicitly define where AI should be applied, set usage expectations by role, and embed AI training into onboarding and workflows.

These enablers are not theoretical. Together, these three traits enable not just tool deployment, but capability transformation. Crucially, these organizations did not wait until every system was optimized or every risk removed. They moved forward with discipline, by testing, learning, and adapting in real time. Their advantages compound daily, and they continue to see real throughput gains and higher returns on AI investments.

Blueprint for Operationalizing AI Engineering

As software teams transition from AI-assisted coding to agentic development, the complexity and autonomy of AI participation will increase. This creates new coordination demands, where code may be written, reviewed, or executed by agents working in parallel with humans.

To manage this shift, high-performing companies are investing in centralized control planes—AI engineering consoles that provide visibility, governance, and performance tracking across human and agent workflows.

To scale team-level gains across the organization, leaders must address five key enablers that repeatedly show up in successful transformations:

- **Workflow Redesign:** Adapt processes to support AI-augmented collaboration, such as triage prioritization, smaller PR batching, and review routing.
- **Governance & Safety:** Implement policy-based guardrails for code generated by AI, including approval workflows, traceability, and audit trails.
- **Infrastructure Readiness:** Upgrade pipelines, test harnesses, and IDE integrations to accommodate higher code velocity and automated contributions.
- **Workforce Enablement:** Provide training tailored by role and tenure, promote internal champions, and support structured experimentation.
- **Cross-functional Coordination:** Align engineering, product, and platform teams on how AI tools are adopted and evaluated.

The GAINS™ Diagnostic: Measuring AI Maturity

To support structured improvement, Faros AI offers the GAINS™ (Generative AI Impact Net Score) diagnostic. This diagnostic provides a forward-looking, objective way to manage change and sustain momentum across the AI transformation journey.

It evaluates ten dimensions critical to success in AI-augmented engineering: AI adoption, velocity, flow efficiency, quality, safety, satisfaction, onboarding, platform maturity, organizational structure, and strategic alignment.

Used quarterly, GAINS helps engineering leaders turn ambition into measurable, time-bound outcomes.

To learn more about GAINS, visit www.faros.ai/gains

FAROS™

About Faros

Faros is the system for running engineering with AI. We give engineering leaders visibility into how work operates across code, people, and systems, and control over how that work progresses through enforceable workflows and policy. This enables organizations to deploy AI effectively and improve engineering throughput with stronger cost efficiency.