DORA

State of AI-assisted Software Development

2025



Google Cloud

Platinum sponsors

swarmia thoughtworks

Premier research partner

IT REVOLUTION

Gold sponsors

Buildkite CodeRabbit DATADOG Deloitte.

harness JELLYFISH Octopus Deploy OPSERA

Research partners

GitHub GitLab

skillbench Workhelix

WURSTA

Foreword

Thank you for interest in this eBook.

As a Google Cloud partner, we at Wursta are committed to providing you with the resources and insights you need to make informed decisions about your cloud journey.

We believe that this eBook will provide valuable insights into the benefits and capabilities of Google Cloud. Wursta is equipped to support you with all your AI-assisted software development needs.

If you have any questions or would like to discuss your specific requirements, please do not hesitate to contact us.

Sincerely,

Wursta



Learn more at wursta.com

Contents

03
Executive summary

08
Foreword

11
Understanding your software delivery performance

23
AI adoption and use

33
Exploring AI's relationship to key outcomes

49
DORA AI Capabilities Model

65
Platform engineering

73
Value stream management

79
The AI mirror: How AI reflects and amplifies your organization's true capabilities

89
Metrics frameworks

95
Final thoughts: From insight to action

97
Acknowledgements

99
Authors

103
Demographics and firmographics

113
Methodology

131
Our research model and its theory

137
Next steps

138
Appendix

Executive summary

Key takeaway: AI is an amplifier

In 2025, the central question for technology leaders is no longer if they should adopt AI, but how to realize its value. DORA's research includes more than 100 hours of qualitative data and survey responses from nearly 5,000 technology professionals from around the world.[1] The research reveals a critical truth: AI's primary role in software development is that of an amplifier. It magnifies the strengths of high-performing organizations and the dysfunctions of struggling ones.

The greatest returns on AI investment come not from the tools themselves, but from a strategic focus on the underlying organizational system: the quality of the internal platform, the clarity of workflows, and the alignment of teams. Without this foundation, AI creates localized pockets of productivity that are often lost to downstream chaos.



Key findings

Drawing on qualitative data and a global survey conducted between June 13 and July 21, 2025, this report uncovers several key findings on the state of AI-assisted software development, including:

AI adoption has become nearly universal. The majority of survey respondents (90%) use AI as part of their work and believe (more than 80%) it has increased their productivity. Yet a notable portion (30%) currently report little to no trust in the code generated by AI, indicating a need for critical validation skills.

Read more in the AI adoption and use chapter.

Successful AI adoption requires more than just tools. Our new DORA AI Capabilities Model identifies seven foundational practices—including a clear AI policy, a healthy data ecosystem, and a user-centric focus—that are proven to amplify the positive impact of AI on organizational performance.

Read more in the DORA AI Capabilities Model chapter.

AI adoption now improves software delivery throughput, a key shift from last year. However, it still increases delivery instability. This suggests that while teams are adapting for speed, their underlying systems have not yet evolved to safely manage AI-accelerated development.

Read more in the Exploring AI's relationship to key outcomes chapter.

This year's research identifies seven distinct team profiles, from "harmonious high-achievers" to teams caught in a "legacy bottleneck," offering a new framework for targeted improvement.

Read more in the Understanding your software delivery performance chapter.

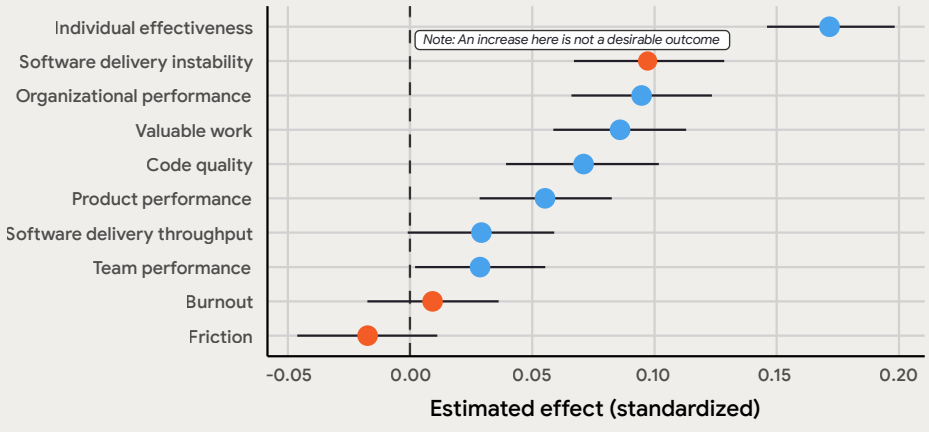
Value stream management (VSM), the practice of visualizing, analyzing, and improving the flow of work from idea to customer, acts as a force multiplier for AI, ensuring that local productivity gains translate into measurable improvements in team and product performance.

Read more in the Value stream management chapter.

90% of organizations have adopted platform engineering, making a high-quality internal platform the essential foundation for AI success.

Read more in the Platform engineering chapter.

The landscape of AI's impact
Estimated effect of AI adoption on key outcomes, with 89% credible intervals



For outcomes in orange, such as Burnout, a negative effect is desirable.
Figure 1: The landscape of AI's impact

Analysis and advice for technology leaders

Successful AI adoption is a systems problem, not a tools problem

Our new DORA AI Capabilities Model reveals that the value of AI is unlocked not by the tools themselves, but by the surrounding technical and cultural environment. We've identified seven foundational capabilities—including a clear AI policy, a healthy data ecosystem, a quality internal platform, and a user-centric focus—that are proven to amplify the positive impact of AI on performance.

Treat your AI adoption as an organizational transformation. The greatest returns will come from investing in the foundational systems that amplify AI's benefits: your internal platform, your data ecosystem, and the core engineering disciplines of your teams. These elements are the essential prerequisites for turning AI's potential into measurable organizational performance.

Broad AI adoption with healthy skepticism

While most developers use AI to increase productivity, there is healthy skepticism about the quality of its output. This "trust but verify" approach is a sign of mature adoption.

The conversation must shift from adoption to effective use. Your training programs should focus on teaching teams how to critically guide, evaluate, and validate AI-generated work, rather than simply encouraging usage.

Seven profiles of team performance

Simple metrics are not enough. We identified seven distinct team profiles, each with a unique combination of performance, stability, and well-being. This model provides a nuanced way to understand your teams' specific challenges and create tailored pathways for improvement.

Use these profiles to diagnose team health beyond software delivery performance metrics. Understand if a team is high-performing but burning out, or stable but stuck on legacy systems, and apply the right interventions.

Quality platforms unlock AI's value

Platform engineering is now nearly universal (90% adoption). Our data shows a direct correlation between a high-quality internal platform and an organization's ability to unlock the value of AI. Organizations that treat their platform as an internal product designed to improve developer experience see significantly greater returns.

Prioritize and fund your platform engineering initiatives. A poor developer experience and fragmented tooling may hamper the impacts of your AI strategy.

A systems view directs AI's potential

This year's research confirms that VSM creates focused improvement, driving higher team and product performance.

VSM acts as a force multiplier for AI investments. By providing a systems-level view, it ensures AI is applied to the right problems, turning localized productivity gains into significant organizational advantages instead of simply creating more downstream chaos.

Using this report

This report details the data behind these findings, including our new DORA AI Capabilities Model, which identifies the key practices that amplify the benefits of AI.

While every organization is unique, our findings provide a framework to inform your strategy and guide your teams. Use this research to form hypotheses, run experiments, and measure the results to discover what drives the highest performance in your specific context.



1. Additional detail about who participated in this year's research is available in the Demographics and firmographics chapter.

DORA COMMUNITY

The **DORA community**[1] provides a platform for professionals to engage with this research and apply it to improve their own organizational performance.



Why join the DORA community?

There are several reasons why you should be a part of the DORA Community:

Learn from experts and peers: The community offers opportunities to learn from guest speakers and other members through presentations and discussions.

Stay up to date with research: Be the first to know about new information and publications from DORA.

Collaborate and discuss: The DORA Community Google Group[2] provides a forum for asynchronous conversations, announcements, and event invitations. This allows members to discuss topics and share their experiences with others in the field.

Engage in community events: A calendar of events, both virtual and in-person, is available on DORA.community.

Contribute to the conversation: Contribute to the conversation by listening, talking, and participating in chats. The community values the input of its members and provides a space for ongoing discussions on topics like leadership, team empowerment, and the evolution of technology practices.

1. The DORA community. https://dora.community
2. The DORA Community Google Group. https://groups.google.com/g/dora-community/about

Foreword

Many believe that the goal of science is to explain the greatest number of observable phenomena with the fewest principles, to confirm deeply-held intuitions, and to reveal surprising insights. For more than a decade, this is exactly what the DORA research program has done.

I'm so excited by how this year's research helps us better understand how we can use AI to improve software.

Gene Kim

Researcher, Vibe Coder, Co-author of *Vibe Coding, The Phoenix Project, DevOps Handbook, Accelerate*

In 2013, I had the privilege of working with Dr. Nicole Forsgren and Jez Humble on the State of DevOps research. This work became the basis for the DevOps Research and Assessment group—DORA—which became part of Google Cloud in 2018.

For many, it's difficult to believe that just over a decade ago, software deployments were dangerous and complex. They required meticulous planning and approvals, and often involved hundreds of risky, error-prone, manual steps. Despite planning and care, deployments still caused massive chaos and disruption, which is why we only dared to do them once per year.

In 2013, the State of DevOps research showed that doing multiple deployments per day was not a crazy idea, and that reliability seemed to require doing smaller deployments far more frequently.

What was even more exciting: you didn't have to be in a startup or in Silicon Valley. You only needed great technical practices (for example, automated builds, automated testing, automated deployments, proactive production telemetry), an architecture that enabled independence of action (the ability to develop, test, and deploy value independently, with little or no coordination cost), and a culture of learning.

Now, 12 years later, as a technology community, we are again faced with a remarkable new technology—AI. And as we did a decade ago, we are asking: Does this new technology actually enable better software delivery and organizational performance?

In 2024, DORA released a landmark report measuring the effects of AI on software delivery performance, one of the first systematic studies of its kind. The results were startling to some. The data suggested that the more AI was used, the worse the software delivery stability and throughput became—the very attributes software development professionals have been working for the last decade to improve

Yes, I've seen and experienced how using AI can lead to problems, everything from silently deleted tests, obviously broken functionality, and even deleted production data. But I've also seen AI used to massively improve outcomes. I started calling last year's report and its findings "the DORA 2024 anomaly"—that is, an exciting mystery to be solved.

This belief was informed by working for the past year with Steve Yegge, famous for his 20 years at Amazon and Google. He chronicled how a memo from Amazon founder Jeff Bezos drove Amazon's transformation from a software monolith into thousands of microservices. This shift helped enable 136,000 deployments deployments per day in 2015, an achievement that for years inspired the DORA research.

Together, Steve and I wrote an upcoming book*, Vibe Coding*, where we define "vibe coding" as any form of coding where you don't type out code by hand. Instead, code emerges from an iterative conversation with an AI.

We describe how vibe coding has changed our lives—it has enabled us to build things we want faster, pursue more ambitious projects, work more autonomously, have more fun, and explore a vastly larger option space (FAAFO!).

Steve and I have seen how using vibe coding can go wrong, resulting in deleted tests, outages, and even deleted code repositories. But we've concluded that this was because the engineering instincts that served us well for decades were now proving woefully insufficient.

Suppose the fastest you've ever traveled is walking at four miles per hour, and someone asks you to drive a car at 50 miles per hour. Without practice and training, you will undoubtedly wreck the car.

We concluded that when AI dramatically accelerates software development, our control systems—that's us—must also speed up.[1] In other words, a decade of DORA research has likely already shown the entire software development industry practices must evolve.

- We need fast feedback loops—faster than ever—to match AI-accelerated code generation.
- We need to work within software architectures that give us independence of action—more than ever, we need to be able to develop, test, and deploy software independently.
- We need a climate for learning, especially given the idiosyncratic nature of AI and its rapid rate of advance.

In *Vibe Coding*, Steve and I included the following case studies that hint at the relevant principles and practices—and why they matter so much in the AI era.

Fast feedback loops and software architecture

Fernando Cornago, global vice-president, Digital and E-Commerce Technology, Adidas, oversees nearly a thousand developers. In their generative AI (gen AI) pilot, they found that teams who worked in loosely coupled architectures and had fast feedback loops "experienced productivity gains of 20% to 30%, as measured by increases in commits, pull requests, and overall feature-delivery velocity," and had a "50% increase in 'Happy Time'"—more hands-on coding and less administrative toil.

In contrast, teams with slower feedback loops due to tight coupling with the Enterprise Resource Planning (ERP) systems saw little or no AI benefits at all.[2]

Culture of learning

We also appreciated the case study from Bruno Passos, group product manager, Developer Experience, Booking.com, which has a team of more than 3,000 developers. In their gen AI innovation efforts, they found that, "developer uptake of vibe coding and coding assistant tools was uneven … Bruno's team soon realized the missing ingredient was training. When developers learned how to give their coding assistant more explicit instructions and more effective context, they found up to 30% increases in merge requests and higher job satisfaction."[3]

Both these case studies point to the exciting possibility that AI amplifies the strengths and weaknesses of our engineering practices. Individuals, teams, and teams of teams with great engineering practices are poised to get outstanding benefits from AI.

We believe that those who don't have such practices will likely have a very bad time, something the "2024 DORA anomaly" hinted at.

I am grateful and honored to have collaborated with Google's DORA team, along with our extended team of experts and researchers whose work and achievements I admire, to help inform this year's research.

What excites me most is the scale of the 2025 research: With nearly 5,000 participants, we'll be able to conduct a survey of practice that will hopefully produce "Eureka!" moments like those of a decade ago. I'm confident we will see similar breakthroughs in the months ahead.

Some of the findings have already made it into the report, but many more tantalizing insights are emerging, and I'm excited to share those findings in the months and years to come.

My gratitude goes to the entire DORA team and the extended contributors who made this groundbreaking research possible.

1. The Nyquist stability criterion from control theory tells us that any control system must operate at least twice as fast as the system it controls.
2. Kim, Gene, and Steve Yegge. *Vibe Coding: Building Production-Grade Software With GenAI, Chat, Agents, and Beyond*. Foreword by Dario Amodei (IT Revolution, 2025), 57.
3. Ibid, 58.



Understanding your software delivery performance: A look at seven team profiles

Nathen Harvey
DORA Lead, Google Cloud

Software delivery performance

The moment when new software has been released is worth celebrating, because its primary value can be determined only once the world can use it. These users may be customers, partners, co-workers, strangers, and even other technology systems.



It's also the moment where we begin to understand how the software will perform in a production environment, and how well it meets the needs of our users. There are many things we can do in advance of this moment to increase our confidence that the software will do what we expect it to, but the moment of release is where the theoretical becomes practical—or not.

Launching new software is more than just launching new applications and services. Once an application has been released, users' feedback will encourage (or force) you to make improvements.

Of course, there are many reasons why you might want or need to change an application, including to remediate security vulnerabilities, improve performance, reduce operating costs, or reduce its carbon footprint. At their heart, these considerations can help both users and the long-term success of the application.

We also need to consider the long-term health and well-being of the teams responsible for the building, deploying, operating, and ongoing support of the application. We need to have the right capabilities and conditions in place that allow these teams to drive successful outcomes in a sustainable manner.

These considerations, coupled with the business imperatives to move faster and drive greater success, have led DORA to use software delivery performance as a focal point for our research.

Software delivery performance factors

DORA's software delivery performance factors take a high-level view of the entire delivery process and focus on two key factors: throughput and instability.

Throughput

Throughput is a measure of how many changes can move through the system over a period of time. Higher throughput means that the system can move more changes through to the production environment.

DORA uses three factors to measure software delivery throughput:

Lead time for changes

The amount of time it takes for a change to go from committed to version control to deployed in production.

Deployment frequency

The number of deployments over a given period or the time between deployments.

Failed deployment recovery time

The time it takes to recover from a deployment that fails and requires immediate intervention.

Instability

Instability is a measure of how well the software deployments go. When deployments go well, teams can confidently push more changes into production and users are less likely to experience issues with the application immediately following a deployment.

DORA uses two factors to measure software delivery instability:

Change fail rate

The ratio of deployments that require immediate intervention following a deployment. Likely resulting in a rollback of the changes or a "hotfix" to quickly remediate any issues.

Rework rate

The ratio of deployments that are unplanned but happen as a result of an incident in production.

Taken together, these two factors for software delivery performance give teams a high-level understanding of their software delivery performance. Measuring these over time provides insight into how software delivery performance is changing. These factors can be used to measure any application or service, regardless of the technology stack, the complexity of the deployment processes, or its end users.

Look beyond software delivery performance

While these five metrics provide a vital snapshot of performance, they are ultimately outcomes. They tell you what is happening, but they don't explain why. A low deployment frequency might be caused by technical debt, bureaucratic processes, or team burnout—and the metrics alone can't distinguish between them.

To connect the performance data to the human experience that drives it, we conducted a cluster analysis. This approach moves beyond isolated numbers to reveal seven common team profiles, each telling a deeper story about the interplay between performance, well-being, and environment.



Finding commonality

We conducted a cluster analysis to understand the human and systemic factors behind software delivery performance and identify common patterns. Our statistical clustering approach revealed seven team types while considering the following factors:

Team performance

This factor measures the perceived effectiveness and collaborative strength of an individual's immediate team.

Product performance

This factor measures the success and quality of the products or services the team is building based on characteristics like helping users accomplish important tasks and keeping information safe, and performance metrics such as latency.

Software delivery throughput

This represents the speed and efficiency of the software delivery process.

Software delivery instability

This captures the quality and reliability of the software delivery process.

Individual effectiveness

This factor captures an individual's self-assessed effectiveness and sense of accomplishment at work.

Valuable work

This measures the self-assessed amount of time an individual spends doing work they feel is valuable and worthwhile.

Friction

This measures the extent to which friction hinders an individual's work. Lower amounts of friction are generally considered to be a positive outcome.

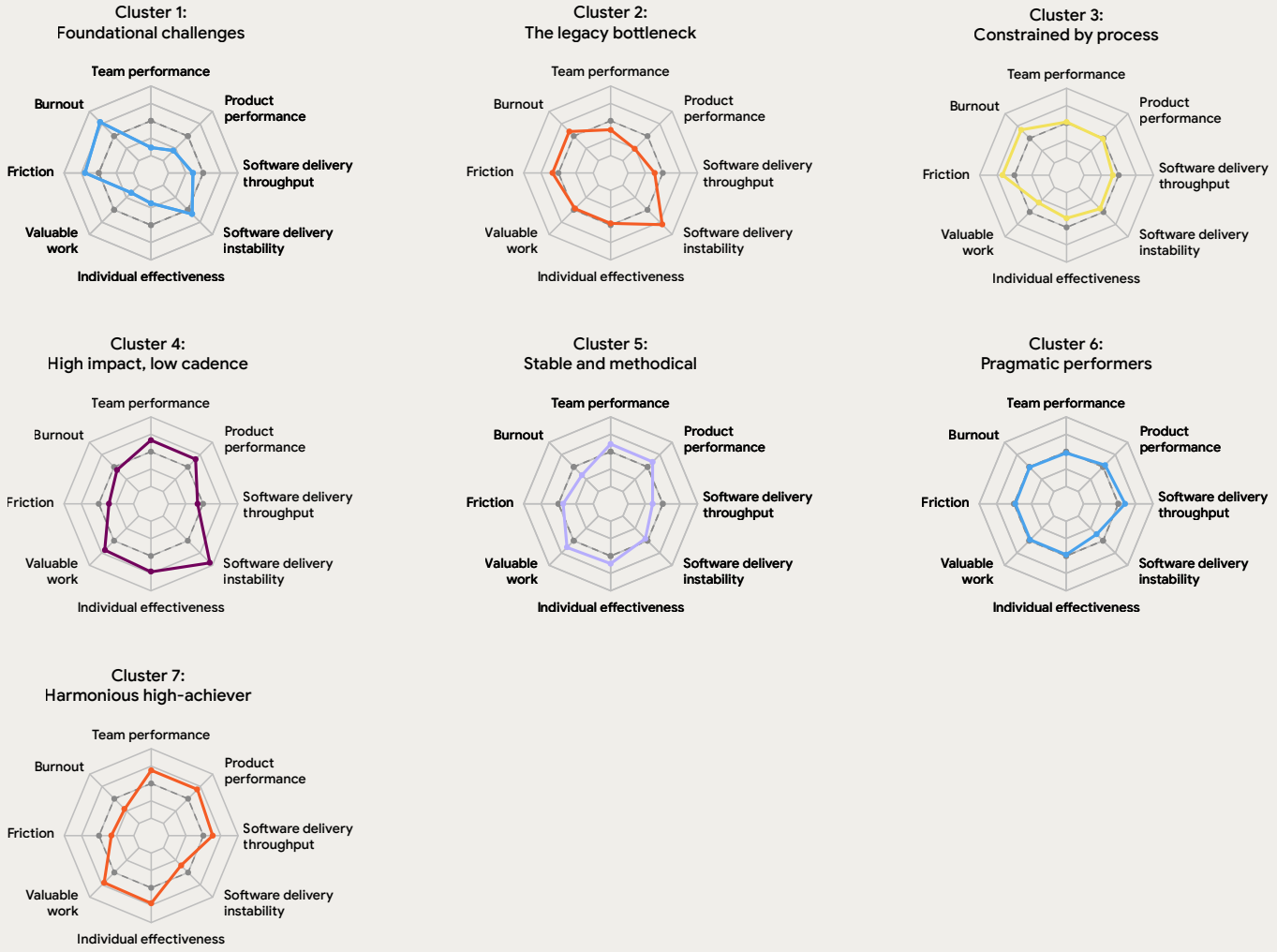
Burnout

This measures feelings of exhaustion and cynicism related to one's work. Lower amounts of burnout are generally considered to be a positive outcome.

Organizations, teams, and individuals usually strive to increase team performance, product performance, software delivery throughput, individual effectiveness, and valuable work while reducing software delivery instability, friction, and burnout.

Our analysis revealed seven distinct team archetypes, ranging from those excelling in healthy, sustainable environments (Harmonious high-achievers) to those trapped by technical debt (Legacy bottleneck) or inefficient processes (Constrained by process).

Performance levels of seven team archetypes



The names and descriptions for each of these clusters are an interpretation of the data. Your team may see similar performance levels as a given cluster but may not feel the cluster name or description describe your team well.

Figure 2: Performance levels of seven team archetypes

Cluster 1: Foundational challenges

These teams are stuck in survival mode, facing significant challenges with fundamental gaps in their processes, environment, and outcomes.

Percentage of respondents: 10% of survey respondents are in cluster 1.

Performance indicators: Key performance indicators related to team output, product delivery, and value creation are consistently low.

Team well-being: The data shows high reported levels of burnout and significant friction.

System stability: There are notable challenges with the stability of the software and operational environment.

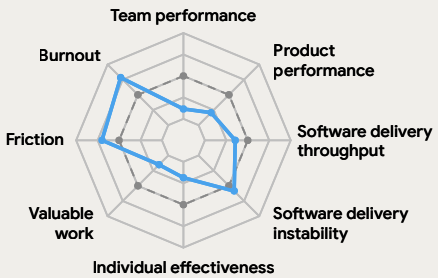


Figure 3: Cluster 1: Foundational challenges

Cluster 2: The legacy bottleneck

Teams in this cluster are in a constant state of reaction, where unstable systems dictate their work and undermine their morale.

Percentage of respondents: 11% of survey respondents are in cluster 2.

Performance indicators: Key metrics for product performance are low. While the team delivers regular updates, the value realized is diminished by ongoing quality issues.

Team well-being: The data indicates a demanding work environment. Team members report elevated levels of friction and burnout.

System stability: There are significant and frequent challenges with the stability of the software and its operational environment, leading to a high volume of unplanned, reactive work.

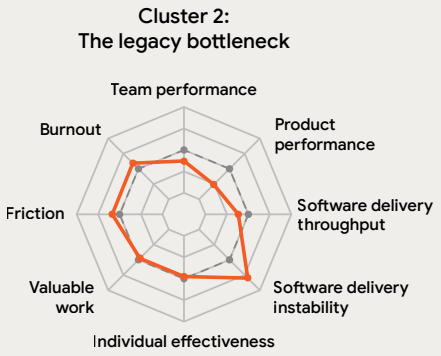


Figure 4: Cluster 2: The legacy bottleneck

Cluster 3: Constrained by process

These teams are running on a treadmill. Despite working on stable systems, their effort is consumed by inefficient processes, leading to high burnout and low impact.

Percentage of respondents: 17% of survey respondents are in cluster 3.

Performance indicators: Key performance indicators show low effectiveness and the creation of limited customer or business value.

Team well-being: The data shows high reported levels of both burnout and friction. This suggests that current workflows and processes are creating a challenging and unsustainable work environment for the team.

System stability: The team's software and operational environments are stable and reliable. This indicates that technical instability is not a primary contributor to the challenges in performance and well-being.

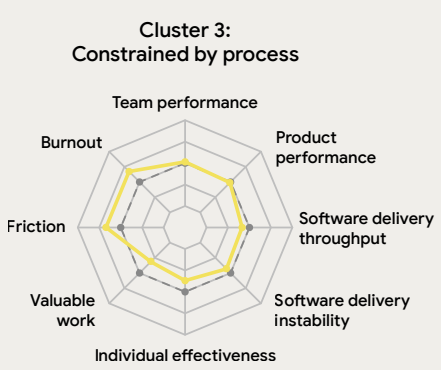


Figure 5: Cluster 3: Constrained by process

Cluster 4: High impact, low cadence

These teams produce high-impact work, reflected in strong product performance and high individual effectiveness. However, this is coupled with a low-cadence delivery model characterized by low software delivery throughput and high instability.

Percentage of respondents: 7% of survey respondents are in cluster 4.

Performance indicators: The team consistently achieves top-tier levels of productivity. Both effectiveness and product performance metrics are strong.

Team well-being: The data indicates a low-friction environment, suggesting that team processes are efficient and collaborative.

System stability: The operational environment is characterized by a high degree of instability. This level of volatility represents a significant risk to service reliability and long-term sustainability.

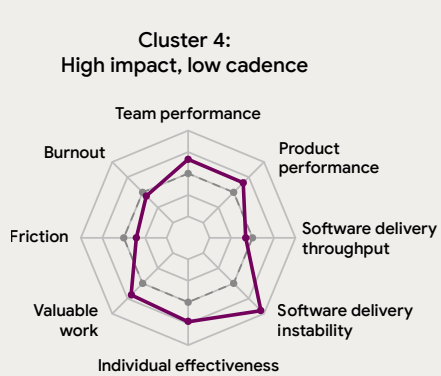


Figure 6: Cluster 4: High impact, low cadence

Cluster 5: Stable and methodical

These teams are the steady artisans of the software world, delivering high-quality, valuable work at a deliberate and sustainable pace.

Percentage of respondents: 15% of survey respondents are in cluster 5.

Performance indicators: Key performance indicators for product quality and value creation are consistently positive. However, the team's software delivery throughput is in a lower percentile, indicating a more deliberate pace of work.

Team well-being: The data shows low reported levels of burnout and friction, which points to a healthy and sustainable team environment.

System stability: The team's software and operational environments are characterized by high stability and reliability.

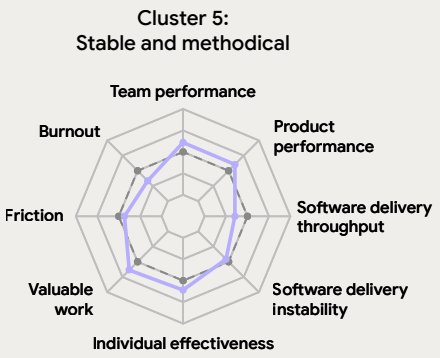


Figure 7: Cluster 5: Stable and methodical

Cluster 6: Pragmatic performers

These teams consistently deliver work with impressive speed and stability, even if their work environment hasn't reached a state of peak engagement.

Percentage of respondents: 20% of survey respondents are in cluster 6.

Performance indicators: Key performance indicators for software delivery are strong, with better-than-average throughput and low instability. The team maintains a steady cadence of valuable output, reliably meeting expectations.

Team well-being: Where this cluster differs from the absolute top tier is in measures of team well-being. The data shows average levels of reported burnout and friction. This indicates a work environment that is functional and sustainable but may lack strong engagement drivers.

System stability: The team's software and operational environments are stable and reliable, providing the solid foundation required for their high performance.

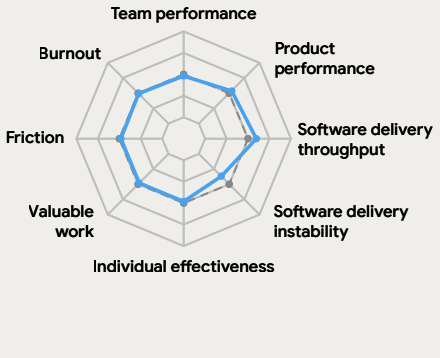


Figure 8: Cluster 6: Pragmatic performers

Cluster 7: Harmonious high-achiever

This is what excellence looks like—a virtuous cycle where a stable, low-friction environment empowers teams to deliver high-quality work sustainably and without burnout.

Percentage of respondents: 20% of survey respondents are in cluster 7.

Performance indicators: The team shows positive metrics across multiple areas, including team well-being, product outcomes, and software delivery.

Team well-being: The work environment is characterized by low reported levels of burnout and friction.

System stability: The team operates on a stable technical foundation that supports both the speed and quality of their work.

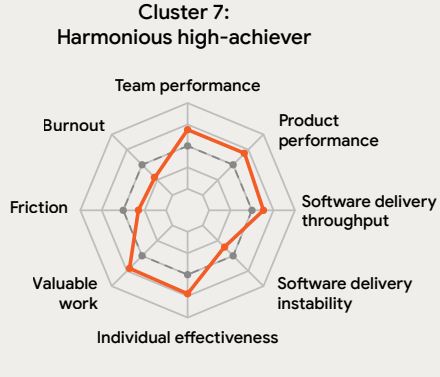
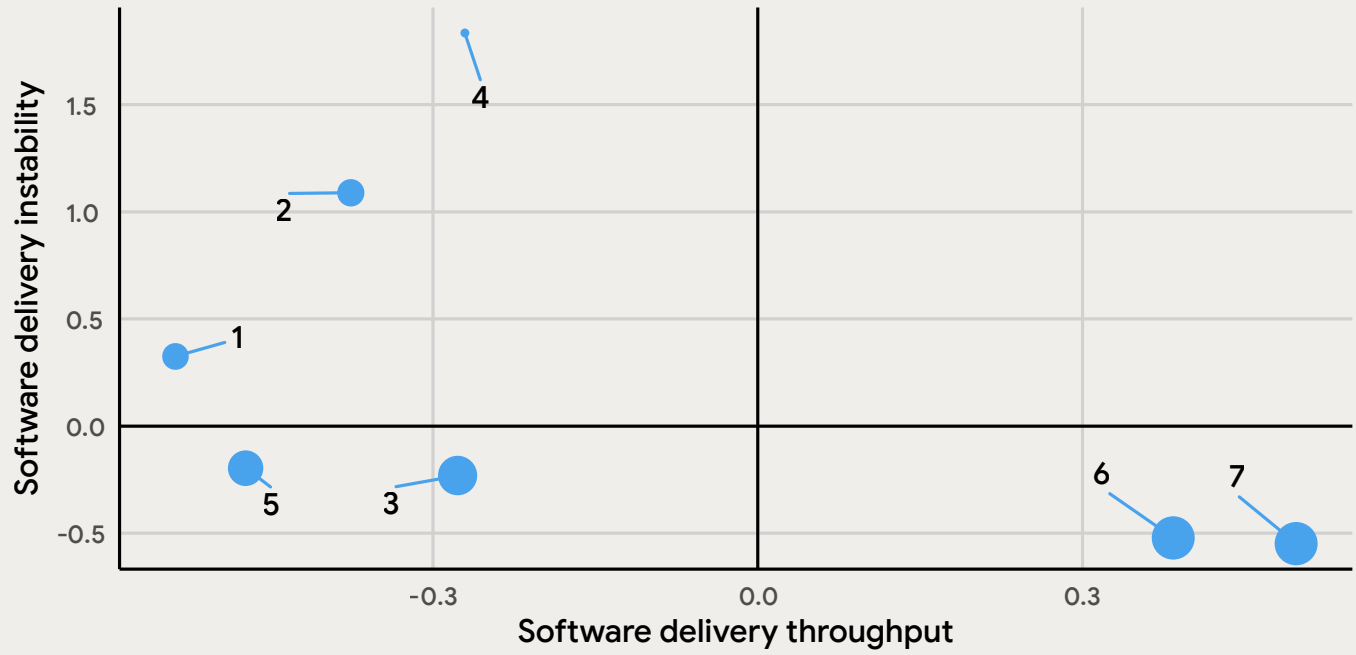


Figure 9: Cluster 7: Harmonious high-achiever

Software delivery performance levels



Size represents total respondents in cluster
Figure 10: Cluster distribution across software delivery performance factors

Figure 10 provides powerful evidence for a core tenet of DORA research, that the "speed vs. stability" trade-off is a myth. The best performers (clusters 6 and 7) excel at both dimensions simultaneously. Conversely, struggles are evident at the other end of the spectrum. Some groups, like those facing foundational challenges (cluster 1), struggle with both throughput and stability, while other high-impact, low-cadence teams (cluster 4) demonstrate that speed without stability is a dangerous and unsustainable proposition.

Excellence is achievable. Clusters 6 and 7 represent nearly 40% of the total sample. Their existence provides an empirical anchor for what is possible—a benchmark that organizations can strive for. While achieving this state is clearly difficult, these groups serve as a powerful testament to the fact that high-velocity, high-quality software delivery is not a theoretical ideal but an observable reality.

How do you compare?

You may be wondering how your team compares to the rest of the participants in this year's research. It is important to remember that these measurements are taken at the application or service level. Doing so encourages cross-functional ownership and accountability for improvement.

Furthermore, the best, most insightful comparisons of software delivery performance are for the same application or service over time. The goal is continuous learning and improvement, not necessarily achieving top software delivery performance.

Figures 11 through 15 provide insight into the distribution of responses we received in the 2025 DORA survey.

Lead time for changes distribution

Survey question:

What is your lead time for changes (that is, how long does it take to go from code committed to code successfully running in production)?

Lead time for changes	% at level	Top %
More than six months	2%	100%
Between one month and six months	13.2%	98%
Between one week and one month	28.3%	84.7%
Between one day and one week	31.9%	56.4%
Less than one day	15%	24.4%
Less than one hour	9.4%	9.4%

Figure 11: Distribution of lead time for changes responses from the 2025 DORA Survey

Deployment frequency

Survey question:

How often does your organization deploy code to production or release it to end users?

Deployment frequency	% at level	Top %
Fewer than once per six months	3.6%	100%
Between once per month and once every six moths	20.3%	96.4%
Between once per week and once per month	31.5%	76.1%
Between once per day and once per week	21.9%	44.6%
Between once per hour and once per day	6.5%	22.7%
On demand (multiple deploys per day)	16.2%	16.2%

Figure 12: Distribution of deployment frequency responses from the 2025 DORA Survey

Failed deployment recovery time distribution

Survey question:

How long does it generally take to restore service after a change to production or release to users results in degraded service (for example, leads to service impairment or service outage) and subsequently requires remediation (for example, requires a hotfix, rollback, fix forward, or patch)?

Failed deployment recovery time	% at level	Top %
More than six months	1%	100%
Between one month and six months	4.9%	98.8%
Between one week and one month	9.4%	93.9%
Between one day and one week	28%	84.5%
Less than one day	35.3%	56.5%
Less than one hour	21.3%	21.3%

Figure 13: Distribution of failed deployment recovery time responses from the 2025 DORA Survey

Change failure rate distribution

Survey question:

Approximately what percentage of changes to production or releases to users result in degraded service (for example, lead to service impairment or service outage) and subsequently require remediation (for example, require a hotfix, rollback, fix forward or patch), if at all?

Change failure rate	% at level	Top %
0%-2%	8.5%	8.5%
2%-4%	8.1%	16.7%
4%-8%	19.6%	36.2%
8%-16%	26%	62.2%
16%-32%	19.5%	81.6%
32%-64%	12.5%	94.1%
>64%	5.9%	100%

Figure 14: Distribution of change failure rate responses from the 2025 DORA Survey

Rework rate distribution

Survey question:

Approximately what percentage of deployments in the last six months were not planned but were performed to address a user-facing bug in the application?

Rework rate	% at level	Top %
0%-2%	6.9%	6.9%
2%-4%	5.8%	12.8%
4%-8%	13.7%	26.5%
8%-16%	26.1%	52.6%
16%-32%	24.7%	77.3%
32%-64%	15.4%	92.7%
>64%	7.3%	100%

Figure 15: Distribution of rework rate responses from the 2025 DORA Survey



Putting software delivery performance into practice

The software delivery performance metrics provide a high-level view of the entire delivery process. Changes to these metrics over time can provide insight into whether things are improving or deteriorating. Interventions required to change the metrics will likely vary for each application; although some common patterns may emerge, such as long review and approval cycles.

Let's walk through a hypothetical example. During a regular retrospective, a team has a discussion about their software delivery performance. They notice that the lead time for changes for the application they work on is starting to increase.

They may have noticed this by looking at a dashboard, but it's just as likely that they noticed this through simple observations. Precision for these metrics isn't always required, and teams will generally know how they are changing over time.

The team wants to reverse this trend, but to do so needs to understand what might be causing it. This is where additional data might be helpful. The team may look at data from their development systems—like their code repository or an analytics tool—and find that code reviews seem to be taking longer.

The team agrees that this is an area that they'd like to improve and discusses potential interventions, including reprioritizing code reviews as part of their daily work and striving for smaller changes that may be easier to review. With these concrete steps identified, they agree to review change approval times and all of the software delivery metrics in a month's time to see how they've progressed.

Whether your team identifies as "Constrained by process" or "Stable and methodical," the goal is the same: to foster a mindset of continuous improvement that moves you toward a more harmonious and high-achieving state.



AI adoption and use

Kevin M. Storer, Ph.D.

User Experience Researcher, Google Cloud

Derek DeBellis

Quantitative User Experience Researcher, Google Cloud

As we continue to explore AI adoption trends by the software development industry, the use of AI in software development has expanded significantly. In this rapidly evolving space, we strive to develop evidence-based guidance to help the software development community navigate these changes. This year's "State of AI-assisted Software Development" represents our most in-depth analysis of AI-assisted development yet.



Findings

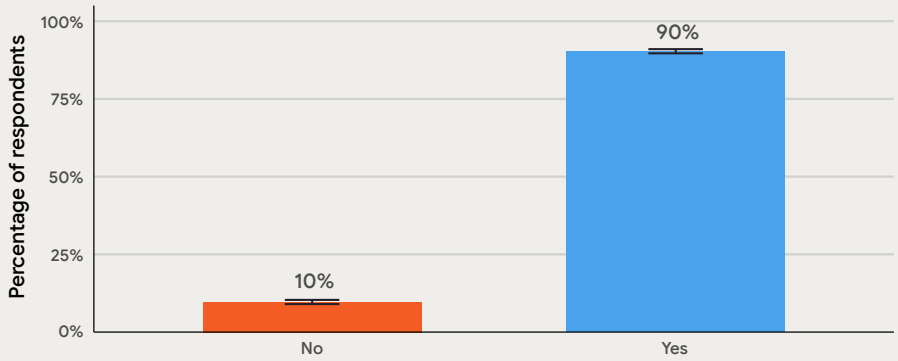
As a whole, our findings regarding the adoption and use of AI by software developers point to a broad adoption of and deep reliance on AI across a diverse range of tasks. This has yielded perceived benefits for both individual productivity and code quality.

Adoption

Ninety percent of this year's survey respondents report using AI at work, a 14.1% increase over the same metric in last year's report. This remarkably high prevalence of AI use at work suggests that AI use in software development has now become the standard.

AI user status

Percentage of respondents who use AI at work



Error bars show the 89% credible interval.
Figure 16: AI user status

"Well, I think over the past year or so, people have realized that generative AI is at the point where it actually works for a lot of things. And now that that is kind of a given, **everyone has applications that could benefit from generative AI in some way.** And so I think there's a lot of motivation to enable new features to save costs on certain things to cut down on the amount of time it takes to create certain things. So I think no one wants to get left behind ... I do think that over time it will become more and more integrated into everything and, **if you're not using generative AI in some way then, yeah, I think it's going to be difficult to keep up."**

Experience

Our respondents report a range of experience using AI tools, with a median of 16 months and a mean of 16.22 months of experience. For reference, ChatGPT was released in November of 2022,[1] approximately 31 months before the launch of our survey. In our data, we have captured both early- and late-adopters in this timeline, and note an observed influx of adoption between late 2023 and mid 2024.

Monthly AI user adoption
Number of new users starting each month, with key industry events

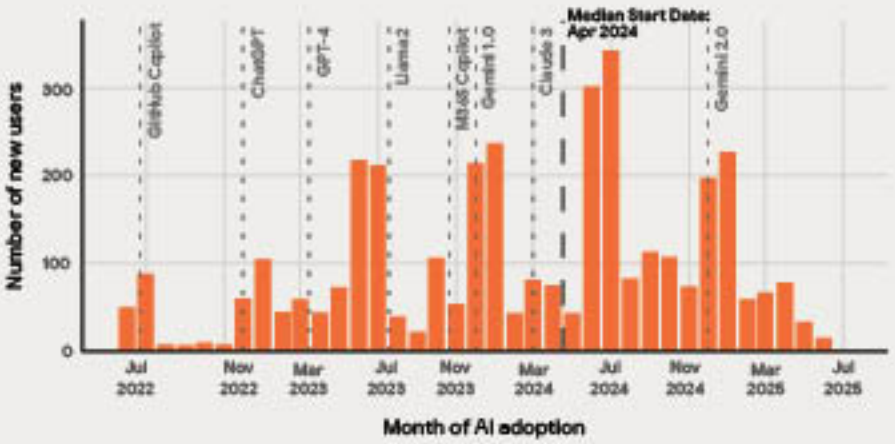


Figure 17: Monthly AI user adoption

Time

AI users in our sample also vary in terms of how much time they spent interacting with AI on their most recent workday. Survey respondents report spending a median of two hours of their most recent workday interacting with AI, representing about one-quarter of an eight-hour workday.

Time spent using AI on a recent workday
Distribution of daily interaction time among AI users

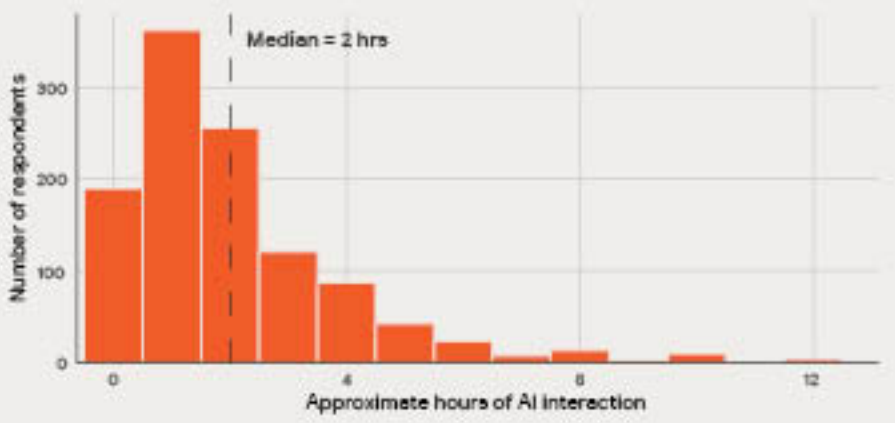


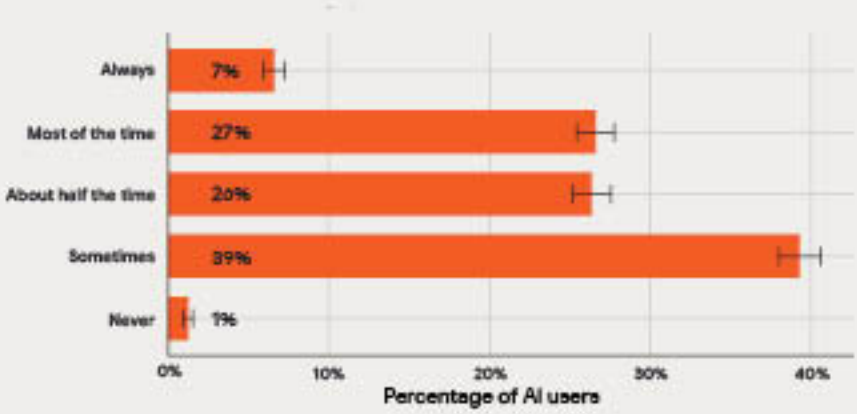
Figure 18: Time spent using AI on a recent workday

Reflexive use

Although AI use is nearly ubiquitous in our sample, reflexive use—the default employment of AI when facing a problem—is not. Among AI users, only 7% report "always" using AI when faced with a problem to solve or a task to complete, while 39% only "sometimes" seek AI for help.

Still, a full 60% of AI users in our survey employ AI "about half the time" or more when encountering a problem to solve or task to complete, suggesting that AI has become a frequent part of the development process.

Reflexive AI use
How often users turn to AI when encountering a problem or task



Error bars show the 89% credible interval.
Figure 19: Reflexive AI use

Sabre

Over the past 18 months, Sabre has actively tracked the adoption of gen AI assistants through usage analytics and satisfaction surveys. Adoption has surged to 74% across developers with varying tenures and experiences, with a notable increase in the use of AI for core development tasks.

This increased usage correlates with higher user satisfaction. A remarkable 86% of users report increased productivity. The steady rise in satisfaction and reported time savings over time suggests that the benefits of gen AI tools grow as users become more proficient.

Our analytics also revealed a slow uptake of the newest gen AI features, like agent mode, with only 25% of users leveraging them. In response, Sabre is enhancing training programs and fostering peer-to-peer knowledge sharing to increase engagement and ensure our teams are proficient with AI tools.

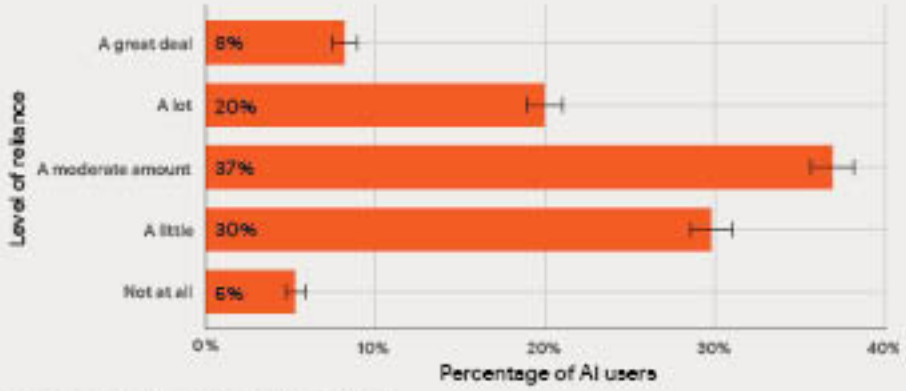
Jacek Ostrowski, VP Platform Engineering, Sabre

Reliance

In addition to using them frequently, we also find development professionals are heavily reliant on AI tools at work.

Only 5% of AI users in our sample report relying on AI at work "not at all", while 65% report relying on AI a "moderate amount" (37%), "a lot" (20%), or a "great deal" (8%).

General reliance on AI at work
Distribution of reliance levels among AI users



Error bars show the 89% credible interval.
Figure 20: General reliance on AI at work

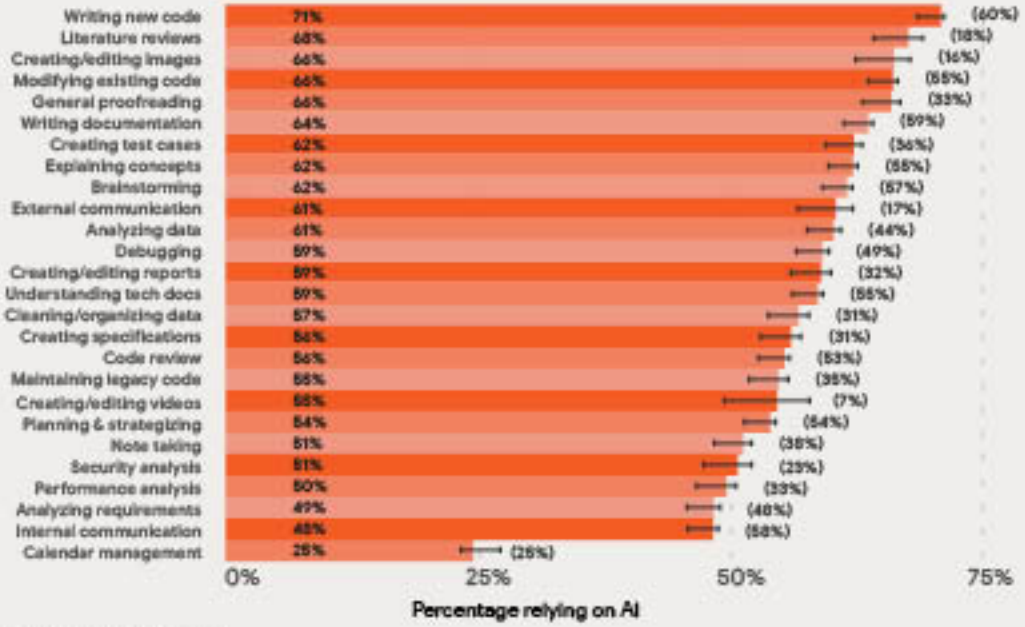
Tasks

As in our 2024 DORA Report,[2] the number one use for AI tools among this year's survey respondents is writing new code, with 71% of respondents who write code using AI to assist them in doing so.

A large majority of respondents whose jobs involve those responsibilities also use AI for literature reviews (68%), modifying existing code (66%), proofreading (66%), and creating or editing images (66%).

Less common uses of AI among respondents whose jobs involve those responsibilities include analyzing requirements (49%), internal communications (48%), and calendar management (25%).

Reliance on AI by task
Percentage of task performers who use AI



Error bars show the 89% credible interval.
Parenthetical is the % of all respondents who perform the task. The 'Other' category is excluded.
Figure 21: Reliance on AI by task

How customization supports developer engagement

Edward Fraser

Graduate student at UC Berkeley's School of Information

What we studied

As AI assistants become more common in development work, a graduate research team at UC Berkeley studied how student developers use AI-powered integrated development environments (IDEs) in practice. Using eye-tracking data and interviews, the team observed how Python developers with between one and five years of experience tackled two short tasks: one involving an unfamiliar library, and another requiring interpretation of a cryptic function.

By applying insights from this study, developers of all experience levels may find ways of working with AI coding assistants that are more attuned to their needs.

Customization as a solution

To reduce friction and better support focused work, developers and teams can customize their AI tools. Most IDEs now offer features like toggling inline suggestions, enabling "on-demand only" modes, or adjusting the style and structure of suggestions.

Repository-level config files and linked documentation can help AI assistants follow established protocols. Experimenting with these settings can align AI behavior with the cognitive demands of different tasks, helping to reduce disruption and increase the usefulness of AI assistants.

When AI gets in the way

While AI coding assistants are designed to save time and reduce effort, a study conducted at UC Berkeley found that they can also introduce friction in some tasks. For example, student developers embraced AI when handling mechanical tasks like writing boilerplate and installing packages, but when deeper understanding was needed, such as interpreting complex code, those same student developers largely ignored AI suggestions.

Eye-tracking revealed less than 1% visual attention on AI chat during interpretive tasks, compared to nearly 19% during the more mechanical ones. The students in this study often chose to complete tasks manually to retain control and comprehension, even ignoring accurate, time-saving suggestions.

Takeaway for teams

To get the most out of AI coding assistants, developers and teams should invest in customization. This study showed that AI may disrupt interpretive tasks by adding cognitive load, especially when developers are trying to make sense of unfamiliar code.

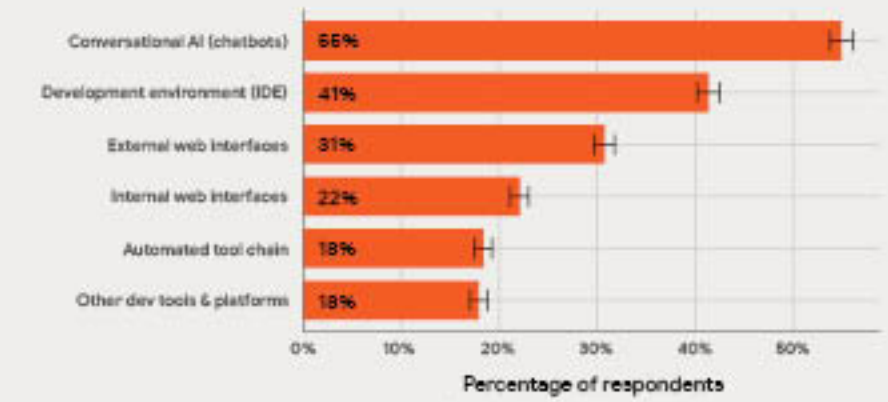
The key is aligning AI support with the nature of the task and preferences of the developer. Tuning your setup can transform AI from a source of friction into a more efficient and satisfying development experience.

Surfaces

Conversational AI chatbots are the most common vehicle for interacting with AI, followed by AI embedded within the IDE.

Respondents report interacting less frequently with AI as part of automated tool chains and other development tools and platforms, but this may be a feature of those AI tools being less visible to their users.

Where people interact with AI
Percentage of respondents using each interaction surface



Error bars show the 89% credible interval. The 'Other' category is excluded.
Figure 22: Where people interact with AI

Mode of AI use

In addition to asking respondents where they interact with AI, and for what purposes, we asked how frequently they interact with AI in each of the following "modes": 1) chat, representing any type of turn-by-turn text-based interaction; 2) predictive text, like tab-to-complete code suggestions; 3) collaborative, using AI to make more broad changes to a codebase; and 4) agent, in which AI is allowed to operate relatively unsupervised and make changes without direct oversight.

Corresponding with chatbots and in-IDE interactions being most frequent among our respondents, textual chat and predictive text modes are the most common modes for respondents to interact with AI. Use of AI agents is least common, with a majority (61%) of respondents reporting "never" interacting with AI tools in an agentic mode. This usage pattern likely reflects the relative maturity of these AI modes; the lower adoption of agentic AI is consistent with its more recent emergence compared to the more established chat and predictive text functionalities.

Frequency of AI interaction modes
Percentage of responses for each interaction mode

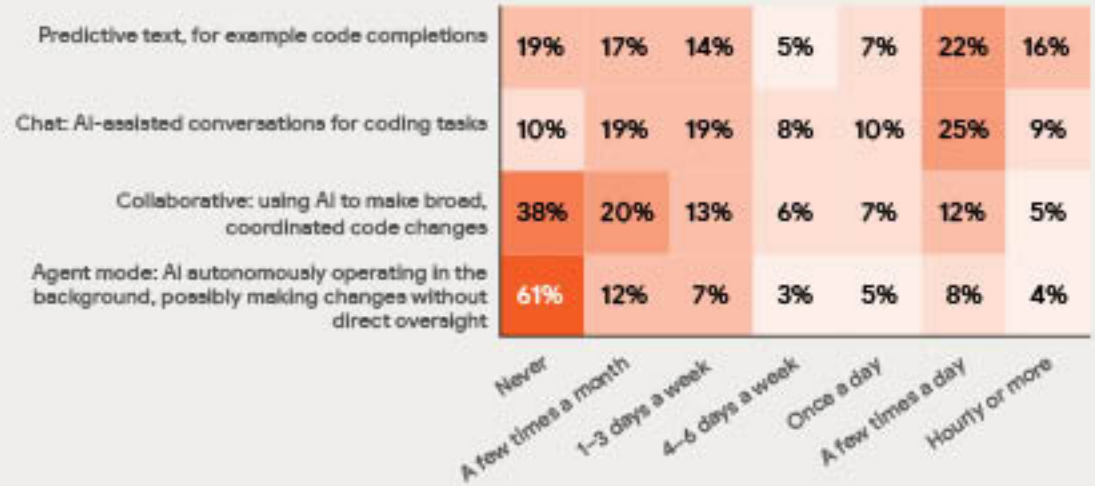
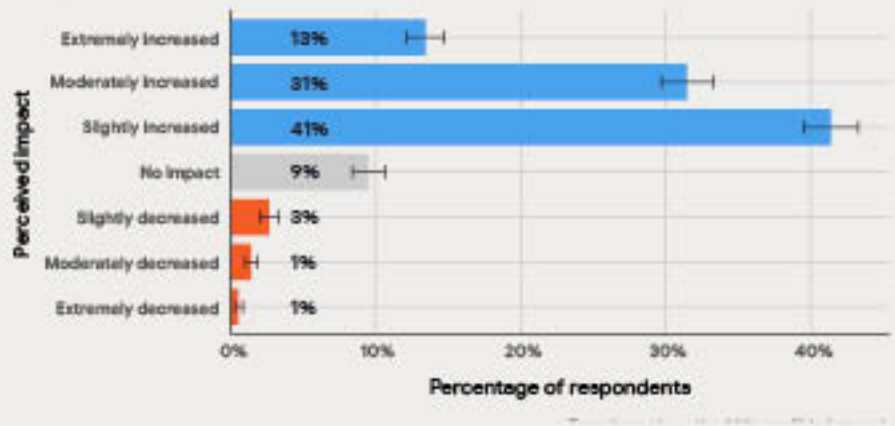


Figure 23: Frequency of AI interaction modes

Individual productivity

More than 80% of this year's survey respondents report a perception that AI has increased their productivity. Although more than 40% report that their productivity has increased only "slightly," fewer than 10% of respondents perceive AI contributing to any decrease in their productivity.

Perceived impact on individual productivity
Respondents' assessment of how AI affects their productivity

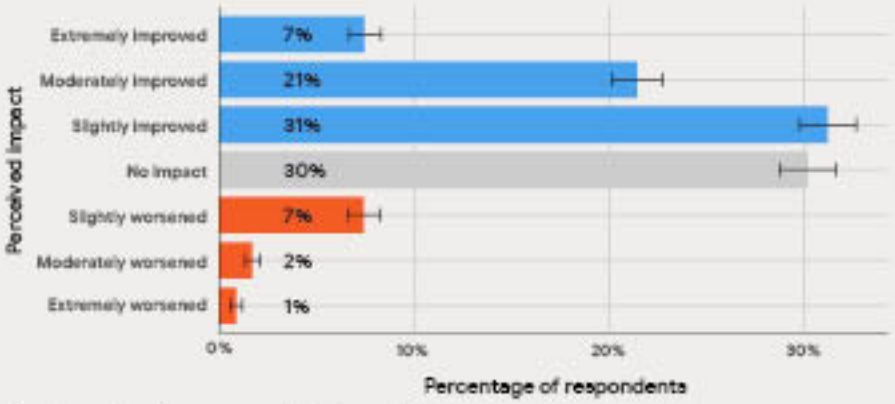


Error bars show the 89% credible interval.
Figure 24: Perceived impact on individual productivity

Code quality

In addition to perceiving positive impacts on their productivity, a majority (59%) of survey respondents also observe that AI has positively impacted their code quality. 31% perceive this increase to be only "slight" and another 30% observe neither positive nor negative impacts. However, just 10% of respondents perceive any negative impacts on their code quality as a result of AI use.

Perceived impact on code quality
Respondents' assessment of how AI affects their code quality



Error bars show the 89% credible interval.
Figure 25: Perceived impact on code quality



"I feel like **AI sometimes writes better code than I do** for certain things, mainly because I feel like it's been trained really well. I mentioned before: code is very binary. It either works or it doesn't. The code that AI writes is usually good enough for my purpose. And **it often uses code standards that I might have accidentally forgotten,** or am too lazy to go back and refactor. So I feel like it kind of creates things more cohesively."

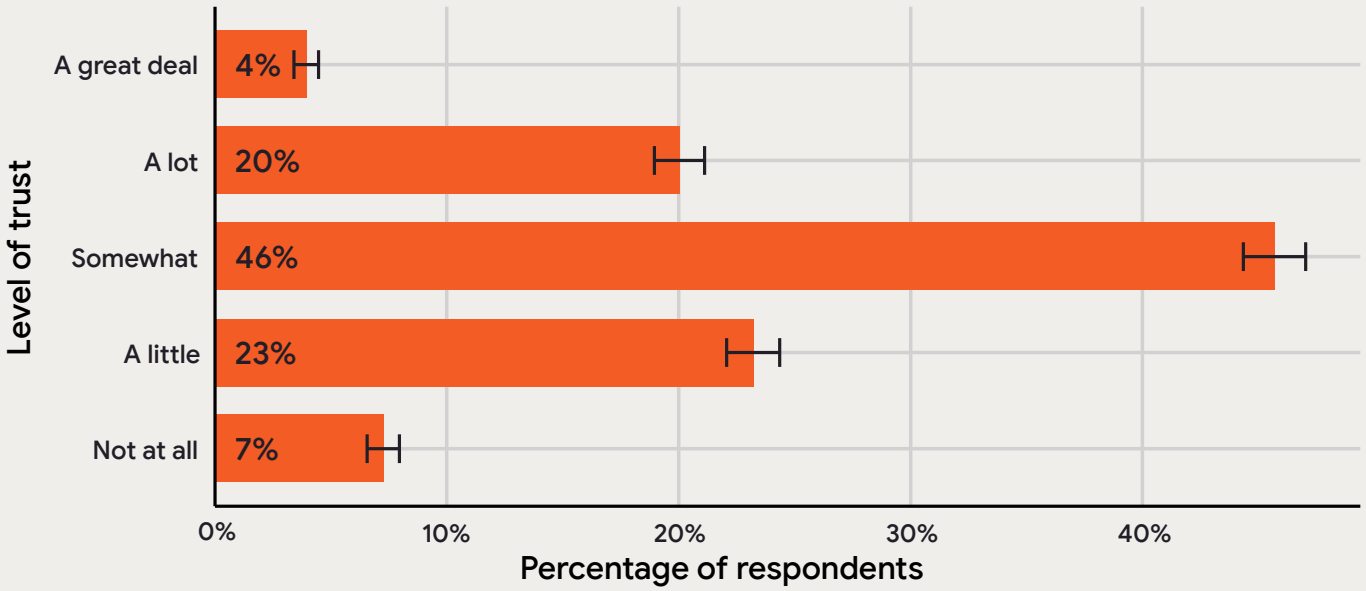
Trust

Similar to the 2024 DORA Report,[2] this year's findings reveal a nuanced landscape of user trust in AI-generated output, with a clear majority of respondents (70%) expressing some degree of confidence in its quality. This includes nearly a quarter of respondents (24%) who report having "a great deal" or "a lot" of trust. While 30% of those surveyed indicate a more reserved stance, with "a little" (23%) or "no trust at all" (7%) in the quality of AI-generated output.

This data highlights a key insight: high levels of AI adoption and perceived benefits can coexist with a measured and nuanced approach to trust. Our findings suggest that absolute trust is not a prerequisite for AI-generated outputs to be useful. This pattern aligns with established behaviors; during our interviews, developers compared this to the healthy skepticism they apply to other widely-used resources, such as solutions found on Stack Overflow, where information is used, but not always implicitly trusted.

The trustworthiness of AI in software development remains an important topic for debate and study, and we have previously identified five strategies to help foster developers' trust in AI.[3] However, our data suggests developers may also already be accounting for this limitation of AI in their work.

Trust in the quality of AI-generated output
Distribution of trust levels among respondents



Error bars show the 89% credible interval.
Figure 26: Trust in the quality of AI-generated output

Final thoughts

Taken together, these findings suggest that the use of AI in software development has become virtually ubiquitous. AI is used in a wide range of development tasks, relied on as part of respondents' workflows, and frequently turned to when facing a problem.

While respondents continue to report concerns about the trustworthiness of AI-generated code, they widely recognize AI's positive impacts on their individual productivity and observed code quality. So despite some imperfections, it seems AI use has rapidly become standard practice for a majority of organizations engaged in software development.

Last year, we found that competitive pressures were a key driver of AI adoption in software development.[2] Many interviewees expressed this as a "fear of missing out" or "getting left behind" by their peer developers and competitor companies.

But, whether social pressure is a logical motivation to adopt a new technology is debatable. While our data shows many positive outcomes of AI adoption, we have also documented notable drawbacks.

For this reason, we caution against interpreting these findings of AI's ubiquity as an indication that all organizations should rapidly move to adopt AI, regardless of their specific needs. Rather, we interpret these findings as a strong signal that everyone engaged in software development—whether an individual contributor, team manager, or executive leader—should think deeply about whether, where, and how AI can and should be applied in their work.

Whether a conservative or permissive approach is right will depend on the context. But, the widespread adoption of AI suggests that organizations can no longer ignore the impacts of its use.

We understand that decisions about the extent to which AI should be integrated into software development are difficult and best made from data. So, in our chapter on DORA's new AI Capabilities Model, we explore how cultural and technological capabilities in organizations affect the outcomes of their AI adoption efforts, to provide insight into ways organizations who choose to integrate AI into their processes can do so successfully.

1. Introducing ChatGPT | OpenAI. https://openai.com/index/chatgpt
2. Accelerate State of DevOps 2024. https://dora.dev/dora-report-2024
3. Fostering developers' trust in generative artificial intelligence. https://dora.dev/research/ai/trust-in-ai

Exploring AI's relationship to key outcomes

Derek DeBellis

Quantitative User Experience Researcher,
Google Cloud





AI is the new normal in software development

In just three years, AI adoption and use has undergone a remarkable shift. A developer using AI may have been surprising in 2022, but today 90% of technology professionals use AI at work. The AI adoption and use chapter shows near-universal adoption, and the trend holds far beyond our data.

The 2025 Stack Overflow Developer Survey found that 84% of developers are now using or planning to use AI tools in their development process, a significant jump from 76% the previous year.[1]

Daily use is also common, with 47% of respondents using AI tools every day. Bolstering this trend, Atlassian's 2025 State of DevEx Survey reports that nearly all developers (99%) now save time using AI tools.[2]

The individual rush is mirrored in corporate strategy. 88% of business leaders reported that accelerating AI adoption is a priority, according to a 2025 report from LinkedIn Corporate Communications.[3] A McKinsey survey found 78% of respondents report that their organizations were using AI regularly for at least one business function.[4]

This priority is reflected in spending, too: Stanford's 2025 HAI AI Index reports that total global corporate investment in AI hit $252.3 billion in 2024, a 26% increase from the previous year.[5] Perhaps nothing illustrates this new reality more starkly than the skills companies are hiring for: U.S. job postings mentioning generative AI skills grew by 323% from 2023 alone.[6]



What is the impact of AI adoption?

Given this rush to adopt, it's important that organizations understand whether there is a corresponding rush in benefits. Widespread adoption doesn't automatically equal widespread value. We need to acknowledge that adoption can be messy, driven as much by hype and fear of missing out (FOMO) as by informed strategy.

Adoption can also be limited and constrained by organizational systems, as we concluded in our 2024 DORA Report, which found that AI returned a lot of promising results but also increased software delivery instability and decreased software delivery throughput.[7] The same 2024 DORA research found an estimated 1.5% reduction in software delivery throughput and an estimated 7.2% increase in software delivery instability for every 25% increase in AI adoption.

Our research is not alone in highlighting these complexities. For instance, a recent study from Model Evaluation & Threat Research (METR) suggested a stark misalignment between perception and reality: developers who were slowed down by AI tools by 19% still believed the tools had made them 20% more efficient.[8] In a similar vein, other third-party research has begun to indicate potential impacts of AI on cognition and well-being, further underscoring that as an industry, we are still in the early stages of understanding the true effects of AI adoption.[9,10,11]

However, when developers were asked to evaluate AI on each area of the SPACE framework,[12] they reported mostly positive impacts and few negative ones.[13] Most respondents in our chapter on AI adoption and use also said that AI has had a positive impact on their code and their productivity.

These mixed signals indicate to us that more evidence-based work should be done to evaluate the true impact of AI on product development, especially given the sheer scale of AI investment and adoption. We believe that the developer community and employers should be setting realistic expectations, and gaining a clear perspective on AI's actual impact is the first step toward managing those expectations responsibly.

Measuring AI adoption

In order to understand how key outcomes differ as a function of AI adoption, we need to be able to measure AI adoption.

This year, we designed our measure of AI adoption to follow some simple rules:

Inclusive: Our measure shouldn't be biased toward any single role (or, indeed, systematically producing higher or lower scores for anything other than AI adoption). For example, a software developer shouldn't automatically score higher just because they use AI for coding. The measure needs to capture a general orientation toward AI, independent of a person's specific job function or anything else besides meaningful AI adoption.

Data-informed: As we do every year, we let the data go first—even if we have strong hypotheses. Part of this process includes conducting an exploratory factor analysis,[14] which is less constrained by our prior beliefs.

Theory-driven: Our conceptualization and measurement of AI adoption should connect with commonly accepted understandings of AI use and our qualitative work.

The result was a factor that was composed of three highly interrelated variables:

Reliance: In the last three months, how much have you relied on AI at work?

Trust: In the last three months, how much did you trust the quality of AI-generated output at work?

Reflexive use: In the last three months, when you encountered a problem to solve or a task to complete at work, how frequently did you use AI?

Items that make up the AI adoption factor

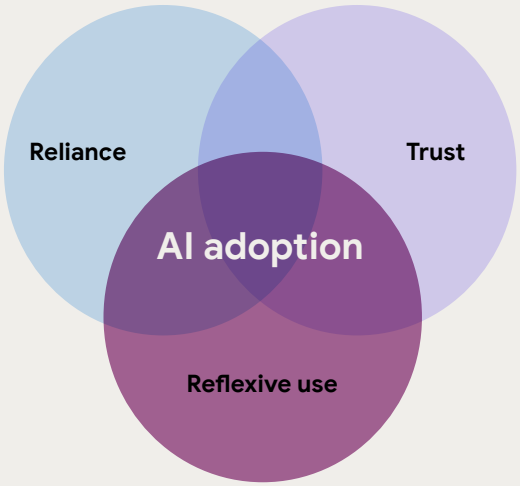


Figure 27: Items that make up the AI adoption factor

The analysis found that these three survey items are answered in a highly similar manner.

This provides evidence that there is a single, underlying construct that is causing these three variables to move together.

We believe this factor captures three conceptually intertwined dimensions: a behavioral dimension (use), a reliance dimension (how deeply AI is woven into an individual's workflow), and a critical attitudinal dimension (trust). This aligns with the literature and our qualitative work.[15,16,17,18]

A feedback loop likely underlies this relationship. Trust is a prerequisite for use, but use is the mechanism for building trust. This creates the powerful feedback loop: as users begin to trust a system enough to use it, their increased usage builds further reliance and deeper trust, creating a cycle of adoption.

Indeed, that cyclical nature makes combining these variables a perfect candidate for a factor, especially given that our survey is a snapshot, not a video of a dynamic process.[19] Adoption is a psychological process involving attitudes, intentions, and actions.

Connecting this measure to outcomes

With a measure of AI adoption established, we can determine whether our outcomes differ when comparing different levels of AI adoption.[20] The basic form is:[21]

When comparing two people who share the same traits, environment, and processes, the person with higher AI adoption will, on average, report {number} more or less {outcome}.[22]

Of course, almost no individual, team, or organization is average, but exploring these average effects can unearth general patterns. These patterns can further help set the context for more nuanced analyses in the DORA AI Capabilities Model chapter, where we explore the conditions under which AI adoption is most (and least) beneficial.

Every year we try to select and construct outcomes that we think represent the goals of people who read the report. In short, we want to evaluate practices in the terms that we think matter to technology professionals. Here are the outcomes we studied this year:

Organizational performance

This is a high-level measure of the overall success of the organization based on characteristics like profitability, market share, and customer satisfaction.

Team performance

This factor measures the perceived effectiveness and collaborative strength of an individual's immediate team.

Product performance

This factor measures the success and quality of the products or services the team is building based on characteristics like helping users accomplish important tasks, keeping information safe, and performance metrics like latency.

Software delivery throughput

This represents the speed and efficiency of the software delivery process. See the Understanding your software delivery performance chapter for more details.

Software delivery instability

This captures the quality and reliability of the software delivery process. See the Understanding your software delivery performance chapter for more details.

Code quality

This captures an individual's assessment of the quality of code underlying the primary application or service they work on.

Individual effectiveness

This factor captures an individual's self-assessed effectiveness and sense of accomplishment at work.

Valuable work

This measures the self-assessed amount of time an individual spends doing work they feel is valuable and worthwhile.

Friction

This measures the extent to which friction hinders an individual's work. Lower amounts of friction are generally considered to be a positive outcome.

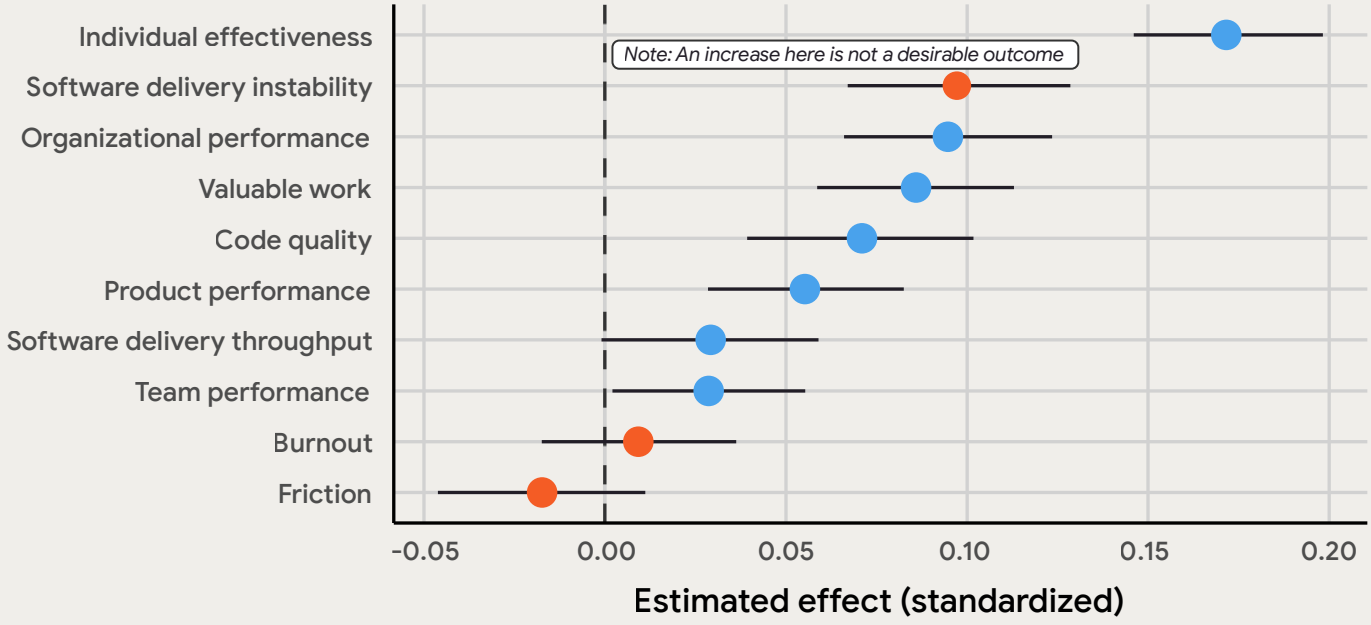
Burnout

This measures feelings of exhaustion and cynicism related to one's work. Lower amounts of burnout are generally considered to be a positive outcome.

The results this year

Figure 28 visualizes the relationships AI adoption has with these outcomes.[23]

The landscape of AI's impact
Estimated effect of AI adoption on key outcomes, with 89% credible intervals



For outcomes in orange, such as Burnout, a negative effect is desirable.
Figure 28: The landscape of AI's impact

We estimate that between two people who share the same traits, environment, and processes, the person with higher AI adoption will report:[24]

- Higher levels of individual effectiveness
- Higher levels of software delivery instability
- Higher levels of organizational performance
- A higher percentage of time doing valuable work
- Higher levels of code quality[25]
- Higher levels of product performance
- Higher levels of software delivery throughput
- Higher levels of team performance
- Similar levels of burnout
- Similar levels of friction

The following sections of this chapter are going to be attempts to make sense of this pattern of results by hypothesizing what their underlying causes are.

To construct these hypotheses, we're going to follow the literature, our qualitative work, our subject-matter experts, and what we've learned from the DORA community. See the Methodology chapter for more details.

The positive associations holding steady since 2024

Since last year's report, several outcomes continue to have positive associations with AI adoption. Let's list them:

- Higher levels of individual effectiveness[26]
- Higher levels of code quality
- Higher levels of team performance
- Higher levels of organizational performance

These steady, positive relationships are becoming a familiar story. We treat them here as a baseline, not a headline, as they are consistent with last year's findings and align with what many practitioners are experiencing. There are likely innumerable underlying mechanisms, many of which may be specific to your organization, team, context, and circumstances.

AI can help individuals by handling boilerplate and other rote scaffolding, surfacing plausible options quickly, providing highly problem-specific output, summarizing and synthesizing large swaths of disparate information, and completing higher-order tasks like design, planning, and analysis. These individual lifts may sum and compound—small wins multiplied across people and cycles—to benefit teams and organizations.

Stubborn results are reminders AI is nested in a larger system

Several outcomes continue to show patterns that suggest a less-than-favorable relationship with AI:

- No relationship with friction
- No relationship with burnout
- AI is associated with an increase in software delivery instability

We think the stubbornness of these effects has more to do with the systems, processes, and culture an individual is nested in. As of now, AI tends to turn up at the keyboard, which can help explain why code quality and productivity benefit, but friction, burnout, and instability may not. They may reside beyond the individual's purview and be tied more directly to how the organization is wired.

We think of these outcomes largely as properties and consequences of the sociotechnical system (combining process and culture). Despite all the benefits, friction remains unaffected, burnout stays flat, and delivery instability rises—unless the surrounding system and culture changes.

Friction

While a tool designed to automate repetitive duties might seem like a clear path to a smoother workflow, our data indicates that workplace friction is a much larger and more complex issue than the mere completion of rote tasks. As we've indicated, some research points to friction as a product of processes beyond the individual.

In 2019, Microsoft identified process issues that get in the way of having a good day:[27]

- Infrastructure issues such as "unstable and slow systems and tools"
- Outdated documentation
- Administrative workload
- Time pressure
- Repetitive tasks

One of their conclusions is that managers should "prioritize and target actions that improve processes and tools."

So, even if AI reduced friction for individual work (for example, writing code), inefficient processes could negate that benefit, especially if they're not prepared to handle the increased volume of changes and the new ways people are trying to work.

For example, an increase in change volume without a corresponding set of evolved guardrails, roles, and "golden paths" could increase verification and coordination costs.[28] However, we don't believe the story of friction is purely systemic.

For an individual, friction doesn't vanish so much as move: It shifts from manual grind to deciding and verifying, possibly in the form of prompt iteration, result vetting, and assessing code that looks remarkably similar to correct code.[29] This could net out to roughly no change in total friction despite being able to produce more impactful outputs and automate certain rote tasks.



"We learned that AI is most effective when it augments the skills of talented engineers. By automating the tedious, repetitive tasks, AI freed up our developers to focus on strategic problem-solving and innovation."

Burnout

While it's tempting to assume that a tool boosting productivity would alleviate burnout, our findings suggest burnout is stubbornly resistant to technological solutions. Burnout is likely heavily influenced by the work culture one is enveloped in.[30]

We've noted this in our own data over the years. Burnout is intricately connected with leadership, priority stability, and generative cultures.[31] A 2017 meta-analysis found some recurring burnout antecedents: low workplace support, a lack of workplace justice, low rewards, job insecurity.[32] Even if AI reduced burnout, the effect would likely be masked by culture's weighty influence.

Further, some clear signals are emerging from our qualitative work which are aligned with literature on work intensification,[33] suggesting perceived capacity gains from AI-assisted development tools have invited higher expectations of work output in some organizations. In these cases, even if AI increases individual effectiveness, the balance between demands and resources remains the same.



"Software development has definitely changed because of AI, and I definitely have felt it since this year. Didn't really feel it last year. But, I think with the recent innovation of MCPs [Model Context Protocol servers], and being able to code with the model, I think **that's really changed a lot [in terms of] releasing features, and the timeline of features, and how much work can be done in a certain duration of time ...** Stakeholders are expecting more work to be done within [the product] in a quicker manner. So **deadlines and projects are on a shorter time crunch, and it's definitely changing the way I work.** So, that worries me a little, because I think they were given a pretty hard deadline from leadership, and by product leadership, in terms of shipping that product."

Software delivery instability

If 11 years of DORA have taught us anything, it's that the technical practices and processes of an organization are intimately tied to software delivery performance. Lacking these foundational capabilities can completely neutralize any gains from AI.

For example, a team that has adopted AI might still experience instability if it hasn't established a strong software delivery pipeline and is highly contingent on other teams to deliver software. Yet our data shows AI adoption not only fails to fix instability, it is currently associated with increasing instability.

Perhaps these technical capabilities are more vital than ever, demanding even stricter adherence to their principles. However, it may be that even this is not enough. Maybe the evidence points toward a more disruptive conclusion: These technical capabilities and measurements no longer suffice. They must evolve for the AI era, be replaced, or be supplemented.

Some might argue that instability is an acceptable trade-off for the gains in development throughput that AI-assisted development enables.

The reasoning is that the volume and speed of AI-assisted delivery could blunt the detrimental effects of instability, perhaps by enabling such rapid bug fixes and updates that the negative impact on the end-user is minimized.

However, when we look beyond pure software delivery metrics, this argument does not hold up. To assess this claim, we checked whether AI adoption weakens the harms of instability on our outcomes which have been hurt historically by instability.

We found no evidence of such a moderating effect. On the contrary, instability still has significant detrimental effects on crucial outcomes like product performance and burnout, which can ultimately negate any perceived gains in throughput.

Changes in last year's patterns suggest adaptation

We have observed some shifts from our 2024 findings:

- AI's relationship with valuable time has reversed from negative to positive
- AI's relationship with software delivery throughput has turned from negative to positive
- AI's relationship with product performance has shifted from neutral to positive

Each of these suggests that people, teams, and tools have adapted. People have had another year to learn how to use AI, organizations and teams have had another year to reconfigure, and AI companies have had another year to develop better models and experiences.[34]

We can start with the tools themselves. Across many benchmarks, AI tools are getting better.[35,36,37] Fine-tuning a pre-trained model on your own data used to be a complex task requiring deep machine learning expertise, but now, many platforms have created streamlined workflows.

Cloud providers have also developed robust tools that allow you to connect your private, proprietary data sources (like a customer database, internal documents, and code repositories) to the fine-tuning process without exposing that data to the public internet or the foundation model provider.

Meanwhile, the ways that organizations use AI have continued to evolve, which might provide AI with some extra capabilities and guardrails for important tasks like code review, test generation, debugging, code refactoring, documentation, and error resolution.

Individuals and teams have likely started to understand where, when, and how AI is most useful. For one, people are likely learning to offload mundane, tedious, and repetitive tasks to AI and spend more time on problem-solving, design, and creative work. This would explain why AI adoption starts to predict, in a reversal of last year's finding, a higher share of time in valuable work.

Indeed, if AI is handling some of the grunt work underlying coding processes (scaffolding, boilerplate, routine transformations), developers may have more time to focus on deploying code, leading to increased software delivery throughput and ultimately to improved product performance.

We could also be observing organizational systems adapting into more fruitful environments for AI, which could empower individuals and teams to get more out of their AI use, and also help their benefits reach teams, products, and organizations.

We explore some potential system constraints that might help explain this in the DORA AI Capabilities Model and The AI mirror chapters.

It's reasonable to wonder why some of these effects were impacted by adaptation and others were not (for example, software delivery instability). The survey data doesn't put us in a good position to answer that question. It is likely an admixture of where people are focusing their efforts, the salience of certain constraints, and the challenge of certain problems. This would likely amount to different learning curves.

"If it helps me, or does something in 30 minutes that is going to take me two hours or more, it's good, because now I have that time and **I can do something else. I can do something different. I can just do more.** And, so, it also helps you, sort of, progress faster in your career as well. Right? Because you are learning new things faster, too."



Conclusion

Across different levels, AI is having a positive impact on most outcomes, with some notable exceptions: It has no measurable relationship with burnout and friction, and it continues its detrimental relationship with software delivery stability.

Comparing some of 2025's findings to last year's, we get a sense that language models, tools, and workflows are evolving along with the people and organizational systems that interact with them. People have found ways to use AI to redirect their efforts to work they consider more valuable, and we're starting to find ways for AI adoption to level up to better delivery throughput and product performance.

It's important to note that AI hasn't made everything better for technology professionals. The stubborn persistence of some issues, including the rise in instability, the flat levels of friction, and burnout, is not entirely a failure of the tool, but also a failure of the system to adapt around it. The persistence of these effects suggests not so much a failure of AI, but more the burden of carrying the weight of the organizational systems one is nested in—and possibly a failure of some of those systems to adapt to the new paradigm.

We believe that the value of AI is not going to be unlocked by the technology itself, but by reimagining the system of work it inhabits. We explore this further in the DORA AI Capabilities Model and The AI mirror chapters.

The sociocognitive impact of AI on professional developers

Daniella Villalba, Ph.D.

User Experience Researcher, Google Cloud

At its core, DORA's focus is the people who develop software. The environment under which developers work plays a critical role in how they experience their work lives. AI is poised to change this environment as organizations reshape their priorities, leaders find new ways to innovate, and AI becomes increasingly integrated into their workflows. AI has the potential to shift the kind of work developers engage in and the problems they work on.

History shows that technological advancements can lead to substantive changes in people's mental models. Before Uber, the idea of paying a stranger for a ride in their personal car was almost unthinkable. Yet today, approximately 31 million people in the U.S. and Canada use the service at least once a month.

Professional developers are at the forefront of this AI-driven transformation, and our goal is to capture the resulting shifts in their experience as they occur.

We chose to investigate six sociocognitive constructs that explore how developers view themselves in relation to their work:

Authentic pride

Meaning of work

Need for cognition

Existential connection

Psychological ownership

Skill reprioritization

Estimated effect of AI adoption score scaled on key outcomes

Points are posterior means; lines are 89% credible intervals

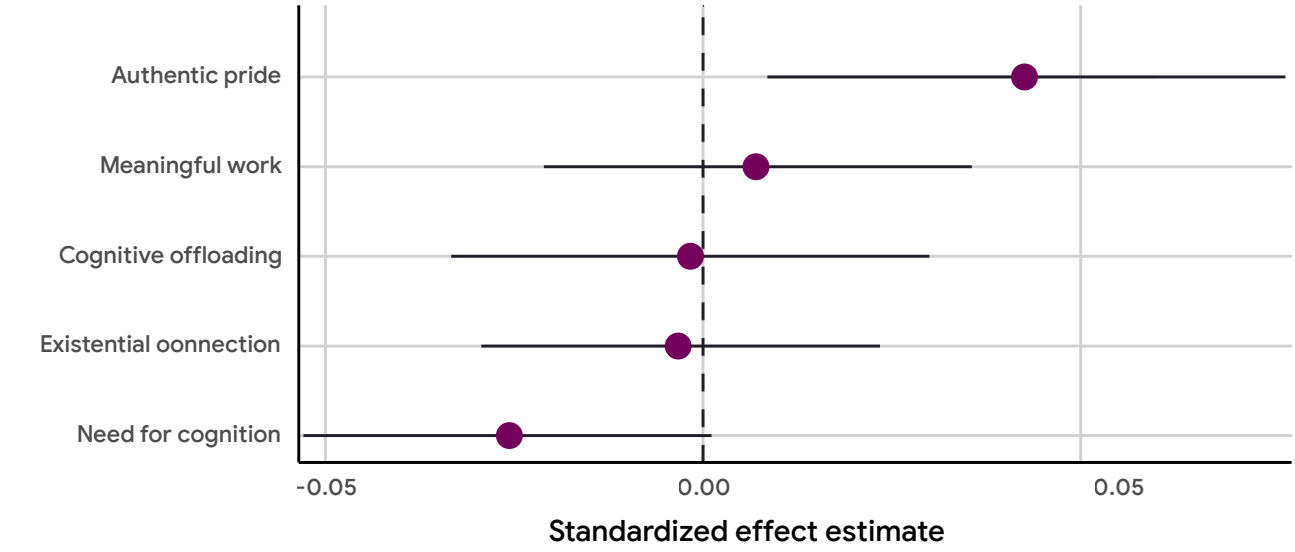


Figure 29: Estimated effect of AI adoption score scaled on key outcomes

Authentic pride

Pride[1] is a basic emotion. We feel pride when we attribute an accomplishment to internal and controllable causes—our behavior. For example, the pride someone might feel after running a marathon because they know all the training miles it took to get there. We feel good about ourselves when we gain mastery or accomplish a difficult task.

Why measure feelings of pride in the context of AI? Because two countervailing hypotheses exist, each with important implications. People might heavily rely on AI and automate their work, leaving little room for effortful achievement. Or, AI might free up people to do work they find valuable and take pride in. We found evidence to support the latter hypothesis: higher AI adoption is associated with greater levels of authentic pride (see Figure 29). This dataset also suggests a clear mechanism: higher levels of AI adoption lead to more time doing valuable work, and people who spend a higher percentage of their time doing work they perceive as valuable report higher feelings of pride.

These findings show the potential downstream benefits of developers learning to offload mundane tasks to AI. They gain control over the most valuable asset—their time—and are free to engage in projects and ideas that matter to them.

Meaningful work

Meaningful work refers to people's desire for their work lives to be spent doing something that matters. Research indicates that people who derive a sense of meaning through their work have higher levels of well-being and job satisfaction.[2]

We wanted to examine whether AI adoption impacted professional developers' perception of whether the work they do has meaning. We hypothesized that AI adoption would either (1) increase developers' sense of meaning in their work by allowing them to automate laborious tasks, and increase the amount of time they spend doing valuable work; or (2) decrease developers' sense of meaning in their work by interfering with their ability to engage in tasks core to their role.

Results indicated no impact from AI on developers' perception of their work as being more or less meaningful. Again, it's possible that we might be too early in this transformation to detect these changes.

Need for cognition

The advent of AI provides people with a quick and easy way to ease their mental load. Some data from academic research has shown that student developers report using AI when they want to "turn off their brain."[3]

While there are people who deeply enjoy engaging in mental activities,[4] it's possible that AI might dampen this enjoyment by providing effortless access to answers. Complex problems can now be solved instantly which may lead to a decreased enjoyment of mental effort for some.

But our findings indicate that AI adoption did not lead to changes in developers' need to engage in mental activities.

Existential connection

Existential connection[5] is the feeling of bridging the gap between your own inner world and someone else's. Philosopher William James noted that this gap makes it hard to truly know another person's experience. This concept captures our ability to form a deep, human link with others, making us feel less alone in our perspectives.

We chose to study developers' existential connections because the rise of AI could change how we interact at work. AI offers instant, personalized answers, which might reduce the need for developers to ask colleagues for help. While efficient, this could lead to fewer conversations and shared problem-solving sessions.

We wondered if relying more on AI and less on each other could weaken workplace relationships and leave people feeling more isolated. Our research, however, found no link between AI adoption and a developer's sense of existential connection.

This could mean it's simply too early to see an impact from this new technology. It's also possible that for every human interaction AI replaces, it creates new opportunities for connection by helping us share and build upon a wider base of collective knowledge.

Psychological ownership

Psychological ownership is the feeling that something is "yours," even if it doesn't actually belong to you. We can have feelings of ownership about tangible objects and intangible constructs, such as our ideas.[6] Many developers feel a sense of ownership about the code they write, and we wondered whether writing code with AI assistance weakens the sense of personal ownership over that code.

Our findings indicate with 78% certainty that AI adoption is not associated with developers feeling a diminished sense of personal ownership over their work. To put it simply, they did not perceive the code as being "less theirs" when they write with AI assistance. This supports the interpretation that today's AI tools function as sophisticated assistants rather than autonomous agents.

Because the AI is seen as a tool to be wielded rather than a collaborator who shares credit, developers have psychologically integrated it into their workflow, much like a compiler or a linter.

However, there is a small (21%) but notable probability that AI decreases a sense of personal ownership (see Figure 30). When developers write code without the assistance of AI, it's clear that they are doing the writing. For some, the injection of AI into their code-writing process could ambiguate the lines between who is doing the writing, reducing their perceived investment and personal control, two key psychological pathways to feeling ownership over an object or task.

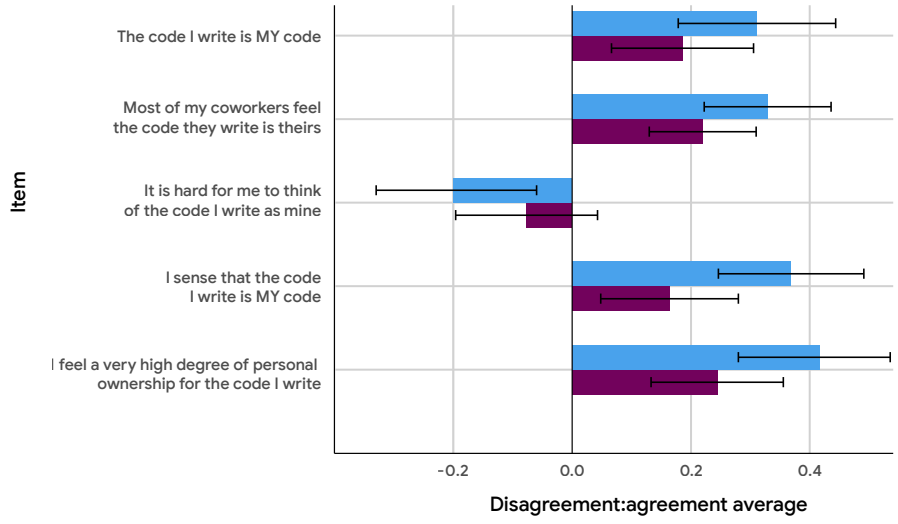


Figure 30: Ownership

Skill reprioritization

As developers increasingly partner with AI, there is a growing conversation about how this collaboration might shift the relative importance of different skills. We wanted to examine whether AI adoption impacted which skills developers think are more or less important for them to do their job.

We hypothesized that AI-specific skills and people-centric skills might be viewed as more important than skills specific to code writing.

We asked participants to rate the following eight skills from most important to least important.

1. Creating technical documentation
2. Problem-solving skills
3. Prompt engineering
4. Programming language syntax memorization
5. Reading and reviewing code
6. Teamwork and collaboration
7. Understanding your team's codebase
8. Writing code

Not surprisingly, AI adoption impacted the perceived importance of prompt engineering. AI adoption also increased the perceived importance of programming language syntax memorization. This finding is interesting and worth further study, as one might expect syntax memorization to be one of the first development-related skills to be perceived as obsolete in the age of AI.

The most surprising finding is that AI adoption did not impact the perceived importance of any other skill.

We are not ready to make conclusions from this data as there are many potential explanations for these findings. These results could signify that developers are in a period of adaptation to new AI-powered workflows, or they may simply reflect a belief that their unique expertise will continue to be indispensable.

Takeaways

Together, these findings indicate that AI adoption has not meaningfully impacted how developers experience their work lives. We will continue proactively monitoring this space for any shifts.

In the meantime, we recommend for organizations to give developers the freedom to double down on work they find valuable. Continue to create opportunities for developers to learn how to leverage AI to their advantage so they can offload toilsome tasks and carve out space in their days to spend more time doing work that matters.

To buffer against potential decreases in psychological ownership, we recommend developers view AI as a tool created to work for them. Even as this technology becomes more autonomous, it is important for developers to see themselves as the ones in the driver's seat.

1. We used a well-established measure of authentic pride. Tracy, Jessica L., Joey T. Cheng, Richard W. Robins, and Kali H. Trzesniewski. "Authentic and hubristic pride: The affective core of self-esteem and narcissism." Self and identity 8, no. 2-3 (2009): 196-213.
2. Steger, Michael F., Bryan J. Dik, and Ryan D. Duffy. "Measuring Meaningful Work: The Work and Meaning Inventory (WAMI)." Journal of Career Assessment, 2012, 20, no. 3. 322–337.
3. Schmidt, Dusana Alshatti, et. al. "Integrating artificial intelligence in higher education: perceptions, challenges, and strategies for academic innovation." Computers and Education Open, vol 9 (2025). https://www.sciencedirect.com/science/article/pii/S2666557325000333
4. Cacioppo, John T., and Richard E. Petty. "The Need for Cognition." Journal of Personality and Social Psychology, 1982, 42, no. 1. 116–131.
5. Pinel, Elizabeth C., Anson E. Long, Erin Q. Murdoch, and Peter Helm. "A prisoner of one's own mind: Identifying and understanding existential isolation." Personality and Individual Differences 105 (2017): 54-63.
6. Peck, Joann, and Suzanne B. Shu. "Psychological Ownership and Feelings of Possession." *The SAGE Handbook of Consumer Psychology*, edited by Curtis P. Haugtvedt et al. (SAGE Publications, 2009), 331–350.

1. 2025 Stack Overflow Developer Survey. https://survey.stackoverflow.co/2025
2. "Atlassian research: AI adoption is rising, but friction persists." https://www.atlassian.com/blog/developer/developer-experience-report-2025
3. "AI Adoption Starts at the Top: 3x more C-suites on LinkedIn are adding AI literacy skills compared to two years ago." https://news.linkedin.com/2025/ai-adoption-starts-at-the-top—3x-more-c-suites-on-linkedin-are-
4. "The state of AI: How organizations are rewiring to capture value." https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai
5. The 2025 AI Index Report. https://hai.stanford.edu/ai-index/2025-ai-index-report, 247. Sourcing quid 2024.
6. Ibid, 228. Sourced from Lightcast data.
7. Accelerate State of DevOps 2024. https://dora.dev/dora-report-2024
8. "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity." https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study
9. "ChatGPT May Be Eroding Critical Thinking Skills, According to a New MIT Study." https://time.com/7295195/ai-chatgpt-google-learning-school
10. Gerlich, Michael. "AI Tools in Society: Impacts on Cognitive Offloading and the Future of Critical Thinking." Societies, 2025, 15, no. 1. Article 6. https://www.mdpi.com/2075-4698/15/1/6
11. "The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers." https://www.microsoft.com/en-us/research/wp-content/uploads/2025/01/lee_2025_ai_critical_thinking_survey.pdf
12. Forsgren, Nicole, Margaret-Anne Storey and Chandra Maddila. "The SPACE of Developer Productivity: There's more to it than you think." https://queue.acm.org/detail.cfm?id=3454124
13. "The SPACE of AI: Real-World Lessons on AI's Impact on Developers." https://arxiv.org/pdf/2508.00178
14. Quantitative Developmental Systems Methodology Core, Penn State. "Intro - Basic Exploratory Factor Analysis." https://quantdev.ssri.psu.edu/tutorials/intro-basic-exploratory-factor-analysis
15. Lee, John D., and Katrina A. See. "Trust in automation: Designing for appropriate reliance." Human factors 46, no. 1 (2004): 50-80.
16. Cody-Allen, Erin, and Rajiv Kishore. "An extension of the UTAUT model with e-quality, trust, and satisfaction constructs." In Proceedings of the 2006 ACM SIGMIS CPR conference on computer personnel research: Forty four years of computer personnel research: achievements, challenges & the future, pp. 82-89. 2006.
17. Reliance is often considered a behavioral manifestation of trust: "Trust in Automation: Integrating Empirical Evidence on Factors that Influence Trust".
18. J. J. Po-An Hsieh and Wei Wang, "Explaining Employees' Extended Use of Complex Information Systems," European Journal of Information Systems 16, no. 3 (2007): 216–27.
19. Adoption is dynamic, but given we have a snapshot, we're treating these highly co-determinant phenomena as unidimensional. If we had panel data, we could be more explicit about the cycle.
20. Last year, we spoke in terms of "effects". This year, however, we will speak in terms of comparisons. Although we try to do the work to create the conditions to speak causally, we don't want to give false assurances that we understand the underlying causal structure. Occasionally we will speak in causal terms, but ultimately, we're doing comparisons. This reasoning is summed up in Regression and Other Stories: "Strictly speaking, though, it is inappropriate to label these as '"effects"' — at least, not without a lot of assumptions … what is observed is an observational pattern … These data allow between-people comparisons … The safest interpretation of a regression is as a comparison … regression is a mathematical tool for making predictions. Regression coefficients can sometimes be interpreted as effects, but they can always be interpreted as average comparisons." Vehtari, Aki, Andrew Gelman and Jennifer Hill, *Regression and Other Stories* (Cambridge University Press, 2020), 84-85.
21. Following the form suggested in *Regression and Other Stories*, 85.
22. Technically a standardized beta weight.
23. Technically, these are standardized beta weights, which equates to the estimated standard deviation difference in the outcome associated with a standard deviation increase in AI adoption.
24. Our dataset is limited, so they're not strictly identical, but they're identical in aspects we consider important to blocking biasing pathways.
25. Bauer, Jared. "Does GitHub Copilot Improve Code Quality? Here's What the Data Says." The GitHub Blog. November 18, 2024. Updated February 6, 2025. https://github.blog/news-insights/research/does-github-copilot-improve-code-quality-heres-what-the-data-says/
26. This was called "productivity" last year. The measure differs slightly and "individual effectiveness" is a more accurate label.
27. Meyer, André N., Earl T. Barr, Christian Bird, and Thomas Zimmermann. "Today was a good day: The daily life of software developers." IEEE Transactions on Software Engineering, 2019, 47, no. 5. (2019): 863—880.
28. Focus time effectiveness of computer assisted protected time for well-being and work engagement of information workers.
29. "Ironies of automation." https://www.sciencedirect.com/science/article/abs/pii/0005109883900468
30. Arnold B. Bakker, Evangelia Demerouti, and Ana I. Sanz-Vergel, "Job Demands–Resources Theory: Ten Years Later," *Annual Review of Organizational Psychology and Organizational Behavior* 10 (2023): 25–53.
31. Accelerate State of DevOps 2024. https://dora.dev/dora-report-2024
32. Aronsson, Gunnar, Töres Theorell, Tom Grape, Anne Hammarström, Christer Hogstedt, Ina Marteinsdottir, Ingmar Skoog, Lil Träskman-Bendz, and Charlotte Hall. "A systematic review including meta-analysis of work environment and burnout symptoms." BMC Public Health, 2017, 17, no. 1. 264.
33. Work intensification: A systematic review of studies from 1989 to 2022.
34. "Technical Performance | The 2025 AI Index Report | Stanford HAI." https://hai.stanford.edu/ai-index/2025-ai-index-report/technical-performance
35. The 2025 AI index Report by Stanford HAI has a lot of benchmark examples from 2023 to 2024. Nature pushes against AI benchmarks: https://www.nature.com/articles/d41586-025-02462-5
36. "Technical Performance | The 2025 AI Index Report | Stanford HAI." https://hai.stanford.edu/ai-index/2025-ai-index-report/technical-performance
37. The Imarena Overview Leaderboard (https://lmarena.ai/leaderboard) shows newer models at the top.



DORA AI Capabilities Model

Kevin M. Storer, Ph.D.

User Experience Researcher, Google Cloud

Derek DeBellis

Quantitative User Experience Researcher, Google Cloud

Nathen Harvey

DORA Lead, Google Cloud

DORA has long strived not just to describe the state of software delivery, but to help organizations make data-backed decisions about how to navigate an ever-changing landscape of development tools, techniques, and technologies. AI is significantly changing software development. While rapid advancements have brought many exciting possibilities, they also bring new questions about how software development might evolve to best meet this moment.

So, this year, we went beyond questions of who is adopting AI and how they're using it, to investigate the conditions in which AI-assisted software developers observe the best outcomes.

We present these findings as our first DORA AI Capabilities Model. The seven AI capabilities in this inaugural model are shown to amplify the benefits of AI adoption. Encompassing both technical and cultural aspects of an organization, our research suggests that investing in developing these areas can help unlock the potential of AI tools.

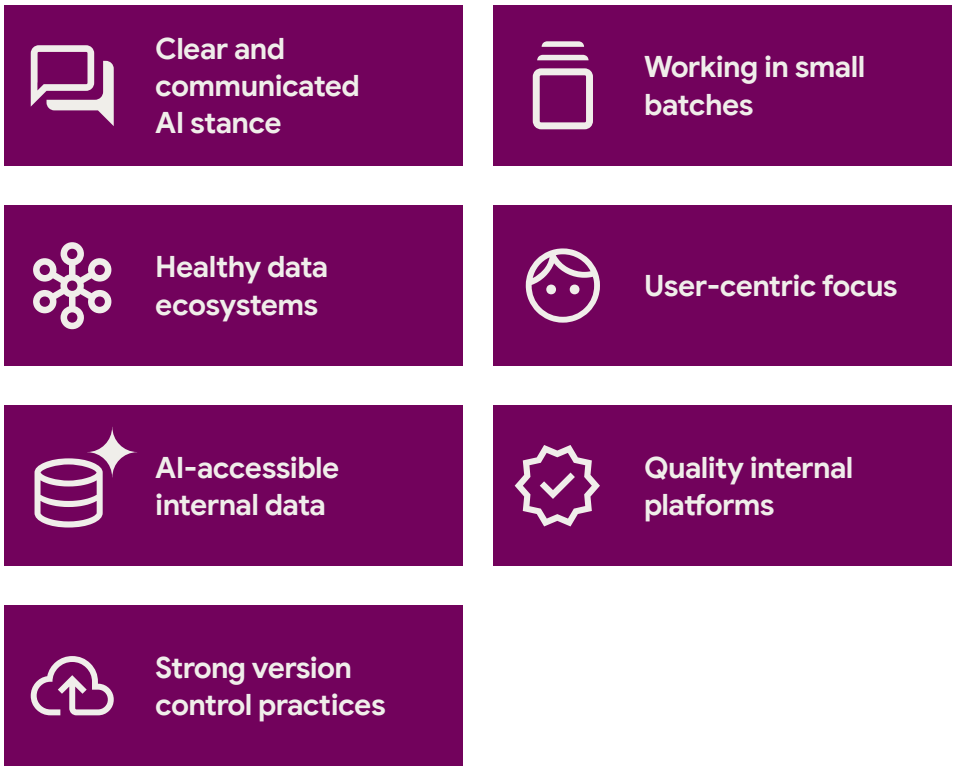
As with the DORA Core Model,[1] we will continue validating, revising, and refining the DORA AI Capabilities Model with further research. We are eager to share future iterations with the DORA Community.

AI capabilities

To develop this model, we hypothesized a wide range of capabilities that might contribute to better outcomes for AI-assisted development teams, based on 78 in-depth interviews, informed opinions from leading subject-matter experts, and previous DORA research.

Through an extensive debate and prioritization process, we selected an initial set of 15 candidate capabilities to include in this year's survey. Of these, a set of seven AI capabilities showed substantial evidence of an interaction with AI use. That is, when teams paired these capabilities with AI adoption, the difference AI made across important outcomes was amplified.

These seven capabilities form the core of our new model:



Clear and communicated AI stance

A "clear and communicated AI stance" refers to the comprehensibility and awareness of an organization's official position on how its developers are expected and permitted to use AI-assisted development tools. Our measure of a clear and communicated AI stance is a single factor, comprised of four individual indicators, measuring respondents' perceptions of:

1. the extent to which AI use feels expected of them at work;
2. the extent to which their organization supports developers experimenting with AI;
3. the extent to which it is clear which AI tools are permitted at work; and
4. the extent to which their organization's AI policy directly applies to them.

In this way, an organization with a clear and communicated AI stance is one that encourages and expects AI use by its developers, supports its developers' experimentation with AI at work, and makes explicit which AI tools are permitted and the applicability of their AI policy for their staff.

A clear and communicated AI stance moderates AI's impact on individual effectiveness

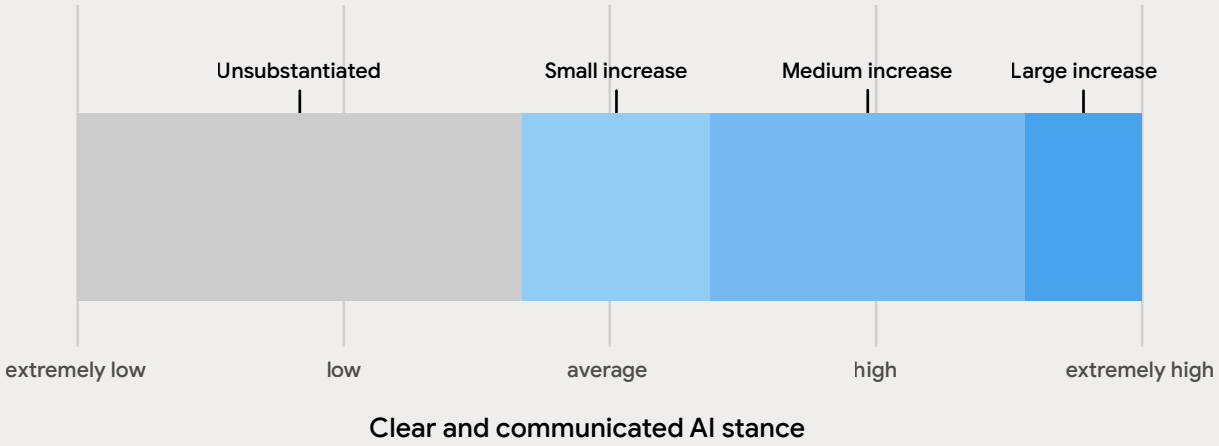


Figure 31: A clear and communicated AI stance moderates AI's impact on individual effectiveness

A clear and communicated AI stance determines AI's impact on organizational performance

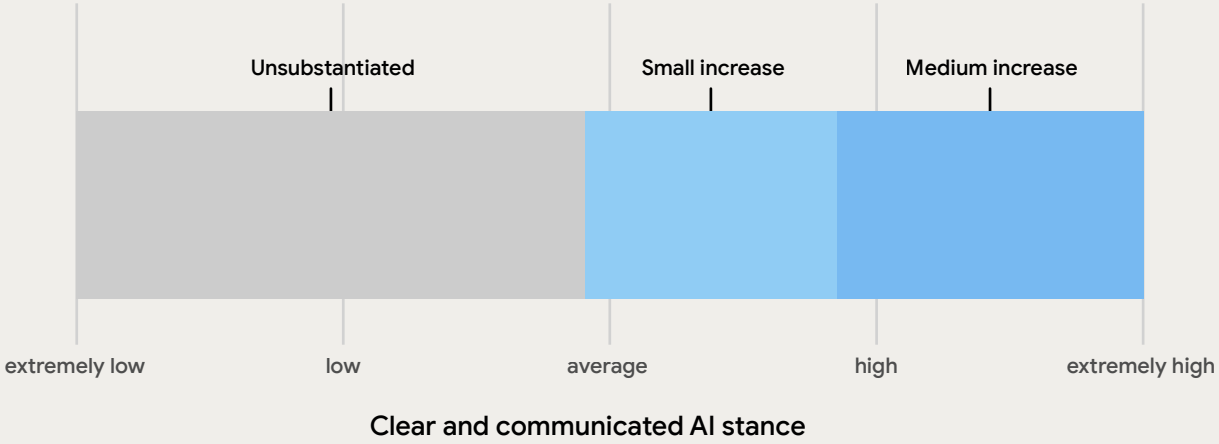


Figure 32: A clear and communicated AI stance determines AI's impact on organizational performanc

A clear and communicated AI stance moderates AI's impact on friction

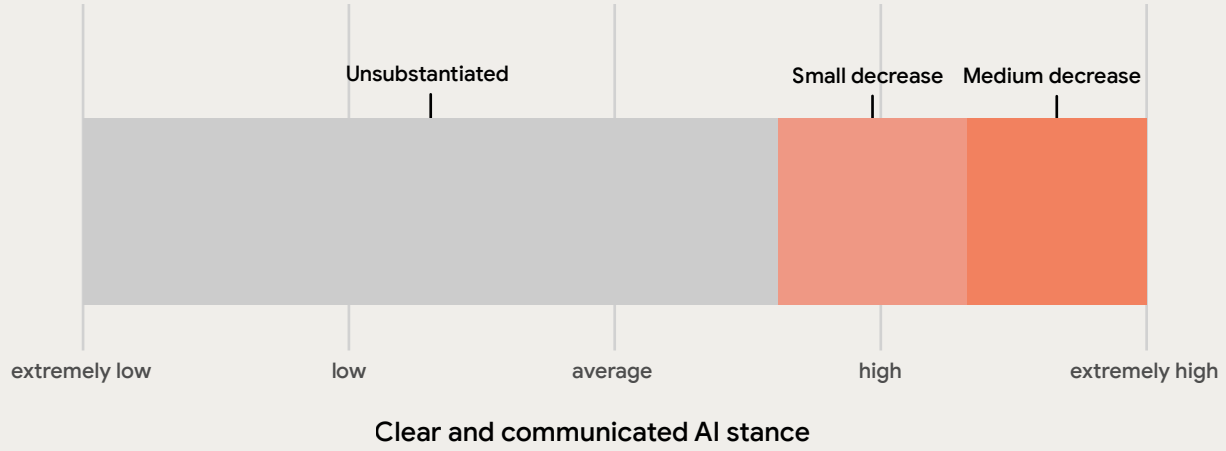


Figure 33: A clear and communicated AI stance moderates AI's impact on friction

A clear and communicated AI stance moderates AI's impact on throughput using AI on a recent workday

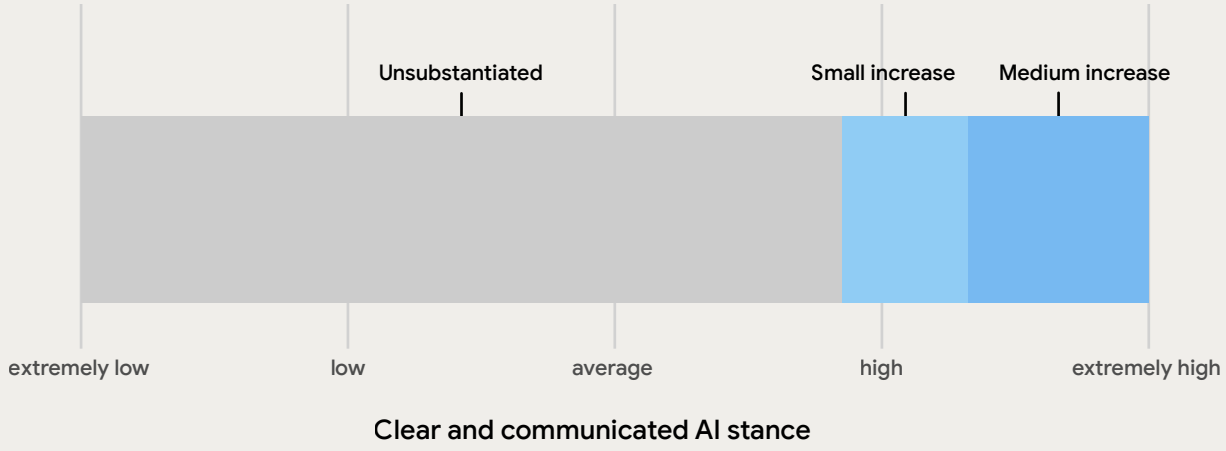


Figure 34: A clear and communicated AI stance moderates AI's impact on throughput using AI on a recent workday

With a high degree of certainty, we found that AI adoption's positive benefits depend on organizations having a clear and communicated AI stance, such that, when they do:

1. AI's positive influence on individual effectiveness is amplified;
2. AI's positive influence on reported organizational performance is amplified; and
3. AI's neutral effect on friction is made beneficial and shown to decrease friction.

With a lesser degree of certainty, we also found that, in the presence of a clear and communicated AI stance:

1. AI's positive influence on software delivery throughput is amplified.

Throughout the in-depth interviews we conducted this year, developers routinely and consistently expressed a lack of clarity and awareness of their organization's stance on AI use in software development. Importantly, this lack of clarity and awareness is likely to manifest in the form of 1) developers who are acting too conservatively, using AI less than they could because they are afraid of overstepping the organization's parameters of acceptable use; and 2) developers who are acting too permissively, using AI in ways that they should not, which do overstep the organization's parameters of acceptable use.

Neither of these cases is optimal.

For this reason, we have previously shared these qualitative insights, concluding that organizations having a clear and communicated stance about the expectations and acceptability of AI in software development can help foster developers' trust in AI,[2] assuage developers' concerns about data privacy in cases where they are unwarranted or resultant from misunderstanding,[3] and scale adoption of AI-assisted development tools across the organization.[4]

These new survey findings affirm our recommendation to invest in making an organization's stance on AI-assisted development clear and communicated to its software developers, and demonstrate measurable positive outcomes of doing so for each individual, team, and organization.

Significantly, this AI capability measures the clarity and awareness—not the specific content—of an organization's stance on AI use in software development. This means that organizations and teams can make their own determinations about what AI stance is appropriate for them, given their unique needs, based on their industry, role, and data infrastructure.

As long as that stance is clearly articulated and widely communicated to their developers, organizations can yield greater positive outcomes from their adoption of AI in their software development processes.

"Why didn't [I] explore [AI] earlier? Some of it's maybe the stigma of 'I don't know how this is going to be looked upon by the other members of my team and management' ... Nobody was talking about it. So, I don't think there was any concern about, 'man, I'm going to get in trouble for this.'

But, it was also, 'I'm not sure how encouraged this is going to be, and if this is something they'll want us to continue doing.' So, I didn't want to necessarily do it in secret either. We do have an AI policy, but it's more about what information we can feed to it in terms of client confidentiality. Those sorts of things. But I think if it were to be encouraged, I might use it more for some more mundane tasks, too."



Healthy data ecosystems

"Healthy data ecosystems" refers to the overall quality of an organization's internal data systems. In our analysis, healthiness of data ecosystems is measured as a single factor, comprised of three individual indicators, measuring respondents' perceptions of:

1. the overall quality of the internal data sources;
2. the accessibility of internal data sources; and
3. the degree to which internal data sources are siloed or divided from one another.

In this way, an organization with a healthy data ecosystem may be understood as an environment in which internal data is high-quality, easily accessible, and unified.

With a high degree of certainty, we found that AI adoption's positive benefits depend on organizations having healthy data ecosystems, such that, when they do, AI's positive influence on organizational performance is amplified.

It is often said that AI models are only as good as the data they train on.

In this case, it appears that this conventional wisdom applies at a local, organizational level.

When organizations invest in creating and maintaining high-quality, accessible, unified data ecosystems, they can yield even higher benefits for their organization's performance than with AI adoption alone.

Data ecosystem health moderates AI's impact on organizational performance

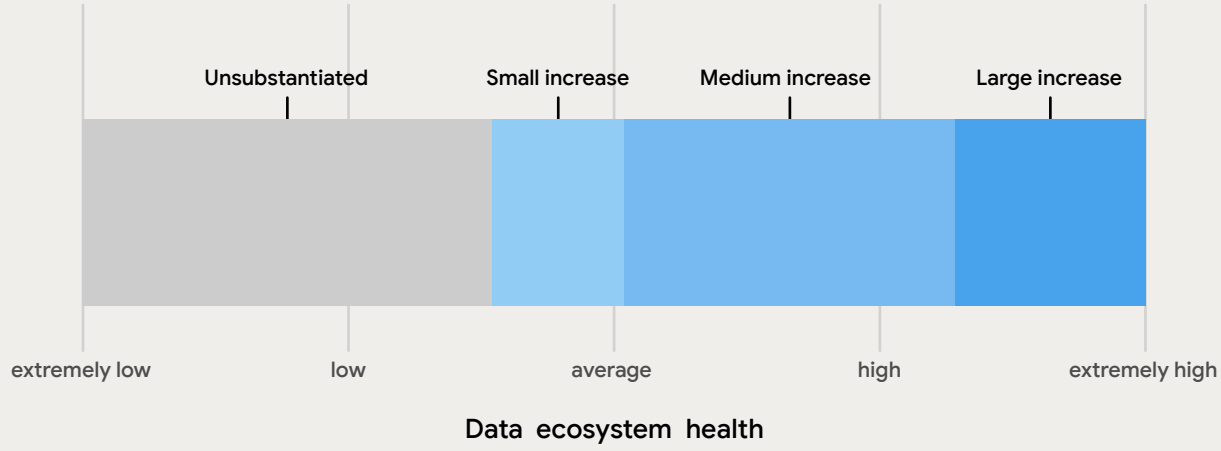


Figure 35: Data ecosystem health moderates AI's impact on organizational performance

AI-accessible internal data

"AI-accessible internal data" refers to the degree to which AI tools are connected to internal organizational data sources and systems. AI-accessible internal data is measured as a single factor, comprised of four individual indicators, measuring respondents':

1. perceptions that AI tools used at work have access to internal company information;
2. perceptions that responses from AI tools used internal company information as context;
3. frequency of inputting internal company information in prompts to AI tools; and
4. frequency of using AI tools to retrieve internal company information.

In this way, an organization with AI-accessible internal data may be understood as one where workers observe that internal data is available to their AI systems and use AI tools to access and process it.

With a high degree of certainty, we found that AI adoption's positive benefits depend on organizations having AI-accessible internal data, such that, when they do:

1. AI's positive influence on individual effectiveness is amplified; and
2. AI's positive influence on code quality is amplified.

While AI tools trained on a general set of knowledge help developers feel more effective and produce higher-quality code, this finding suggests that AI can be even more impactful toward those goals when given access to internal data sources that allow developers to provide their AI tools with company-specific context.

This also suggests that organizations who invest time in connecting their AI tools to their internal systems may observe better outcomes than organizations who rely on the less specialized knowledge provided by generic foundational models.

Put differently, maximizing the individual-effectiveness and code-quality benefits of AI may require a deeper investment than simply procuring AI licenses. In some ways, this finding is unsurprising—if AI can't access internal company data, how useful can it really be?

AI-accessible internal data moderates AI's impact on individual effectiveness

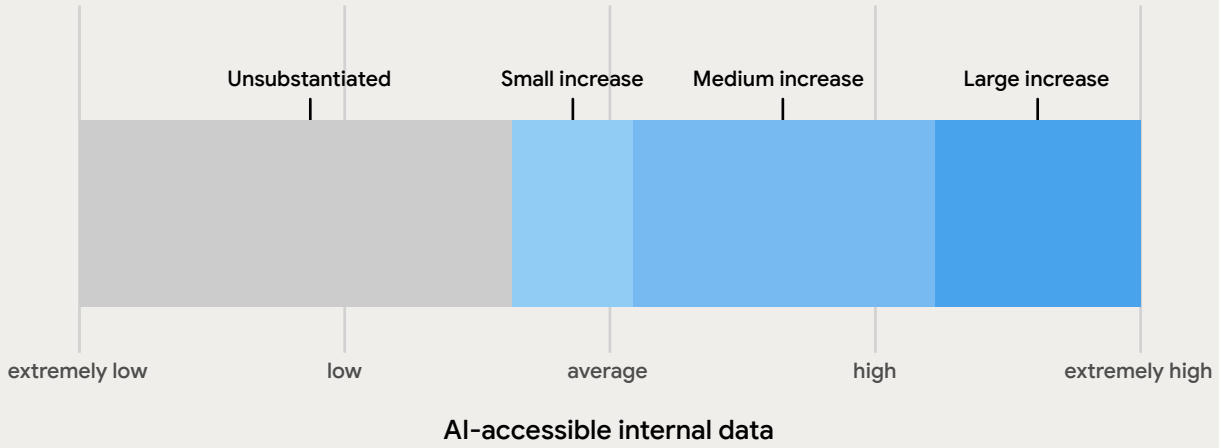


Figure 36: AI-accessible internal data moderates AI's impact on individual effectiveness

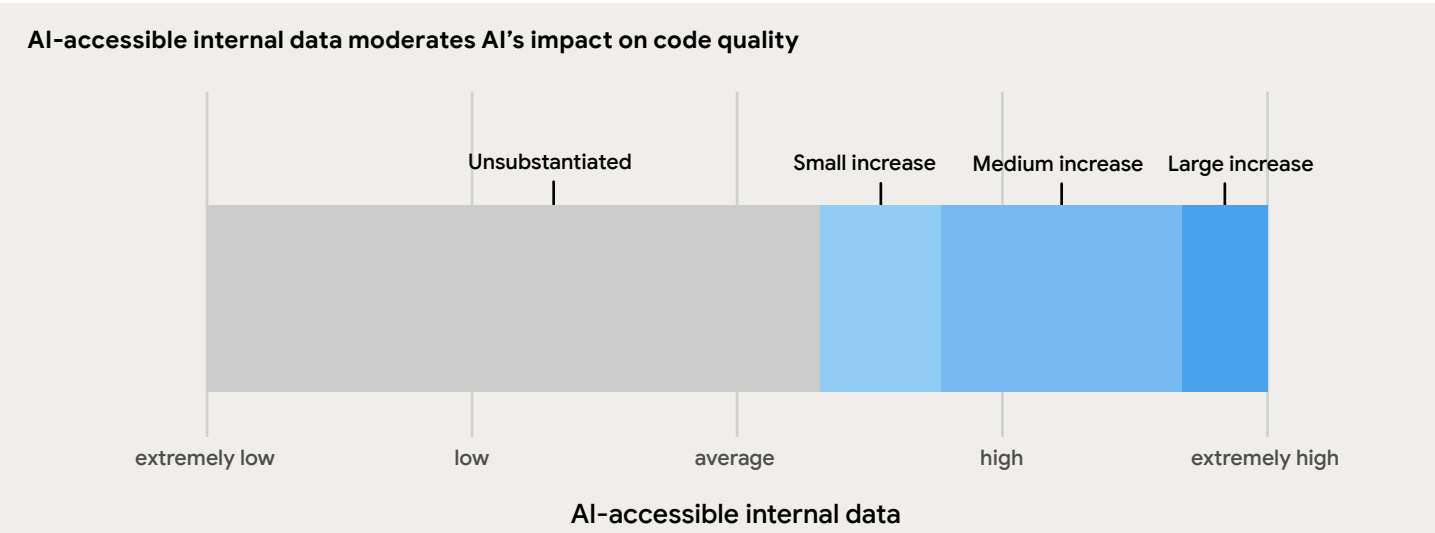


Figure 37: AI-accessible internal data moderates AI's impact on code quality

"I don't think many of my current clients are at a stage where they can actually effectively have any AI system ... they aren't even at a stage where they have their data properly organized ... it's, like, spread out all across the company, and there is no standard system to actually store data, or have it in a standard format. And then, if they want to use AI, there is also a bit—actually not a bit, a lot—of data engineering work actually needed to bring it in a way which can be actually consumed by the gen AI system. I feel many of the companies are not even at that stage where they can actually effectively use that."



Strong version control practices

Strong version control practices have long been foundational to high-performing software development teams. These tools provide a systematic way to manage changes to code and other digital assets over time.

In the age of generative AI, where the volume and velocity of code generation are dramatically increasing, the importance of these practices is amplified. Our research indicates a powerful synergy between mature version control habits and the adoption of AI, highlighting that these practices are crucial for maximizing AI's benefits while mitigating its risks.

With a high degree of certainty, we found that AI adoption's positive benefits depend on respondents' frequency of version control commits. Specifically, in the presence of frequent commits, AI's positive influence on individual effectiveness is amplified.

Additionally, we found that AI adoption's positive benefits depend on respondents' frequency of use of their version control systems' "rollback" features to undo or revert changes. Specifically, in the presence of more frequent rollbacks, AI's positive influence on team performance is amplified.

A key aspect of mature version control is its function as a "psychological safety net." This safety net allows development teams to experiment and innovate with confidence, knowing that they can easily revert to a stable state if something goes wrong. One of the most tangible examples of this is the reliance on rollback or revert features. The ability to undo changes swiftly and without fuss is not just a convenience; it's a critical enabler of speed and resilience.

The rate at which code can be produced by AI may help developers feel more productive. But, as discussed in our chapter Exploring AI's relationship to key outcomes, AI use is also associated with a higher degree of software instability.

We have hypothesized that this is likely, in part, because it is harder to review larger batches of code.

So, although rollback reliance does not directly reduce instability, we suspect that its positive effect on team performance for AI-assisted teams may relate to the importance of being able to rapidly undo changes when working with larger batches of code and the instability that they can produce.

Version control commit frequency moderates AI's impact on individual effect

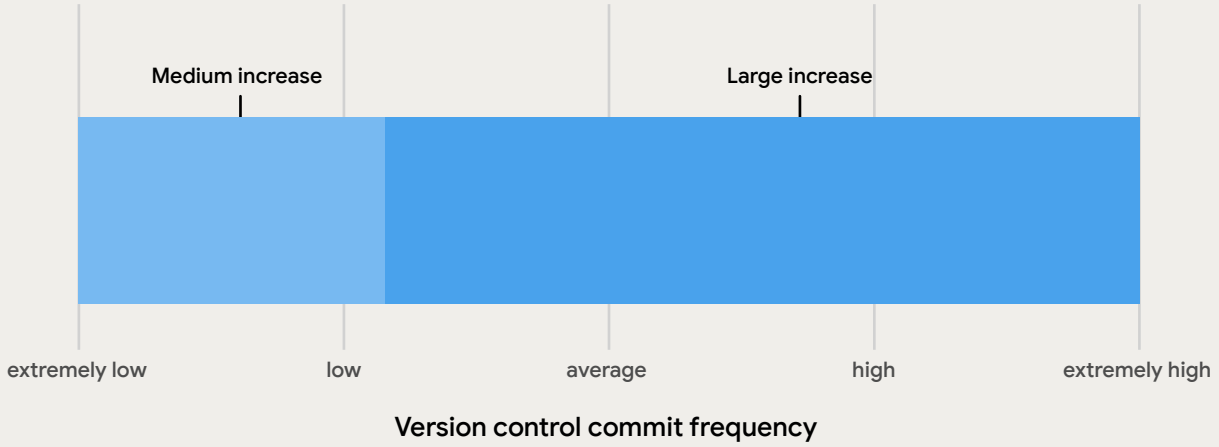


Figure 38: Version control commit frequency moderates AI's impact on individual effect

Ability to rollback moderates AI's impact on team performance

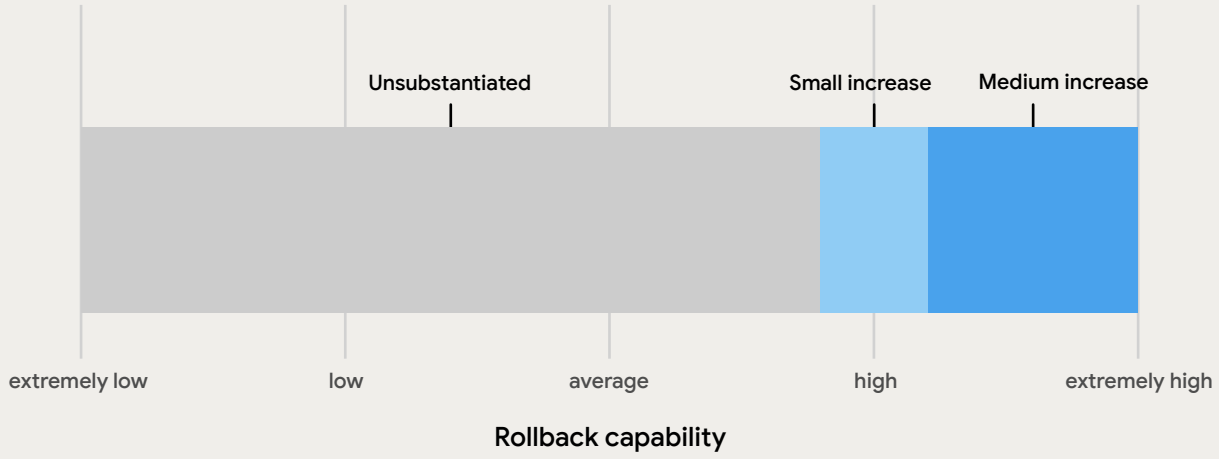


Figure 39: Ability to rollback moderates AI's impact on team performance



Working in small batches

"Working in small batches"[6] is a long-time DORA Capability, which refers to the degree to which teams break down their changes into manageable units that can be quickly tested and evaluated. Working in small batches is measured as a single factor, comprised of three individual indicators, measuring:

1. the approximate number of lines of code committed in the most recent change for respondents' primary application or service;
2. the number of changes typically combined into a single release or deployment; and
3. how long it takes a developer to complete the work assigned in a single task.

A team that scores more highly in terms of working in small batches is one that commits fewer lines of code per change, fewer changes per release, and assigns work that can be completed in a shorter amount of time.

With a high degree of certainty, we found that AI adoption's positive benefits depend on teams working in small batches, such that, when they do:

1. AI's positive influence on product performance is amplified; and
2. AI's neutral effect on friction is made beneficial and shown to decrease friction.

Conversely, we also found that AI adoption's benefits for individual effectiveness are slightly reduced in teams that are working in small batches.

Although these results are mixed, we believe that, overall, they point to a net-positive impact of working in small batches for AI-assisted teams. The observed reduction in reported increases in individual effectiveness from AI use when working in small batches supports our underlying theory that AI predominantly increases perceptions of individual effectiveness by helping developers to quickly generate a large amount of code.

For teams who prioritize working in small batches, it seems natural that observed gains in individual effectiveness would be somewhat less.

More importantly, we argue that individual effectiveness should not necessarily be pursued as a goal in and of itself. Rather, individual effectiveness is a means to realize greater organizational, team, and product performance, and improved developer well-being.

In this case, working in small batches increases reported product performance, while also decreasing perceived friction for AI-assisted teams. We think these benefits outweigh any potential harm to individual effectiveness from working in small batches—in addition to those benefits of working in small batches that have been long-proven as part of our DORA Core Model.[7]

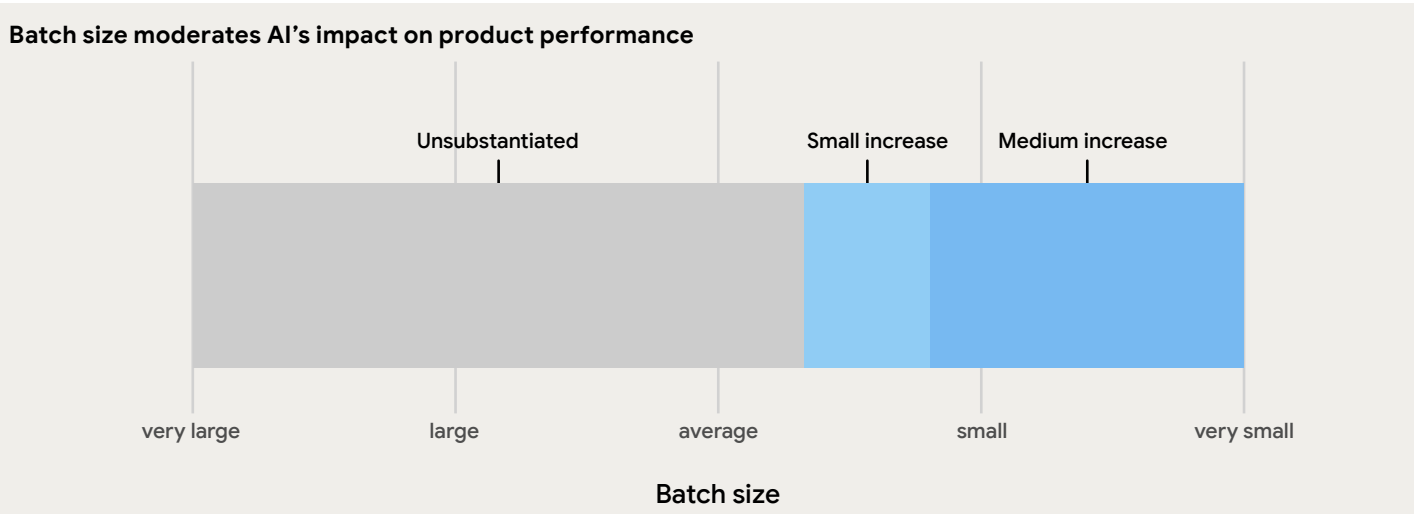


Figure 40: Batch size moderates AI's impact on product performance

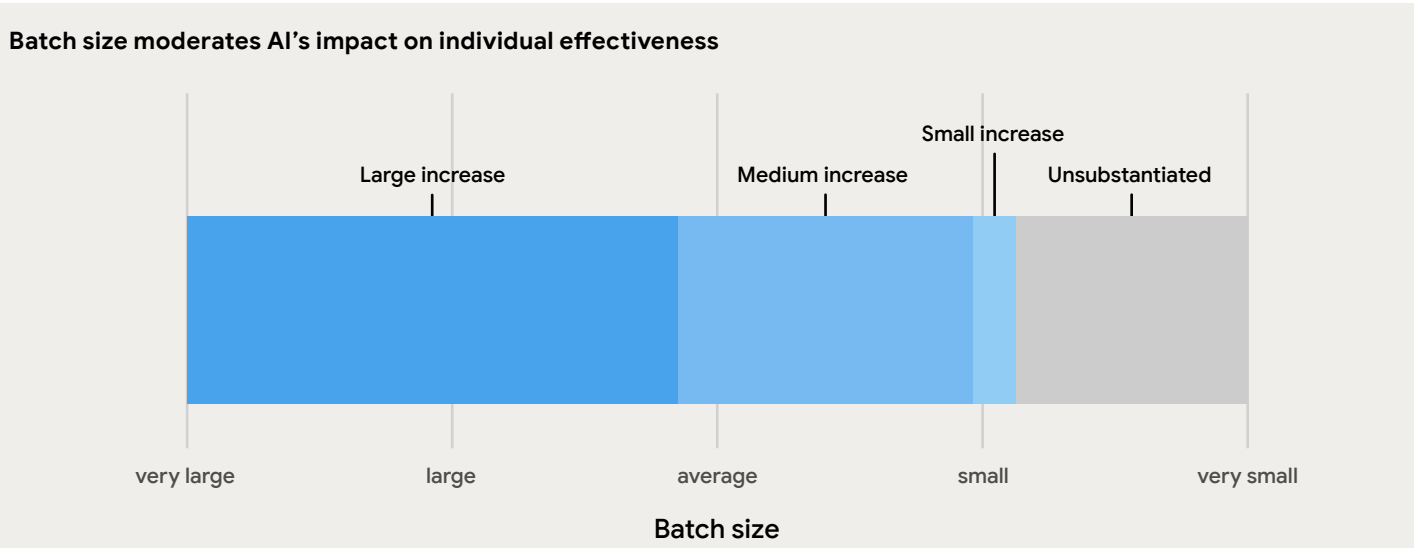


Figure 41: Batch size moderates AI's impact on individual effectiveness

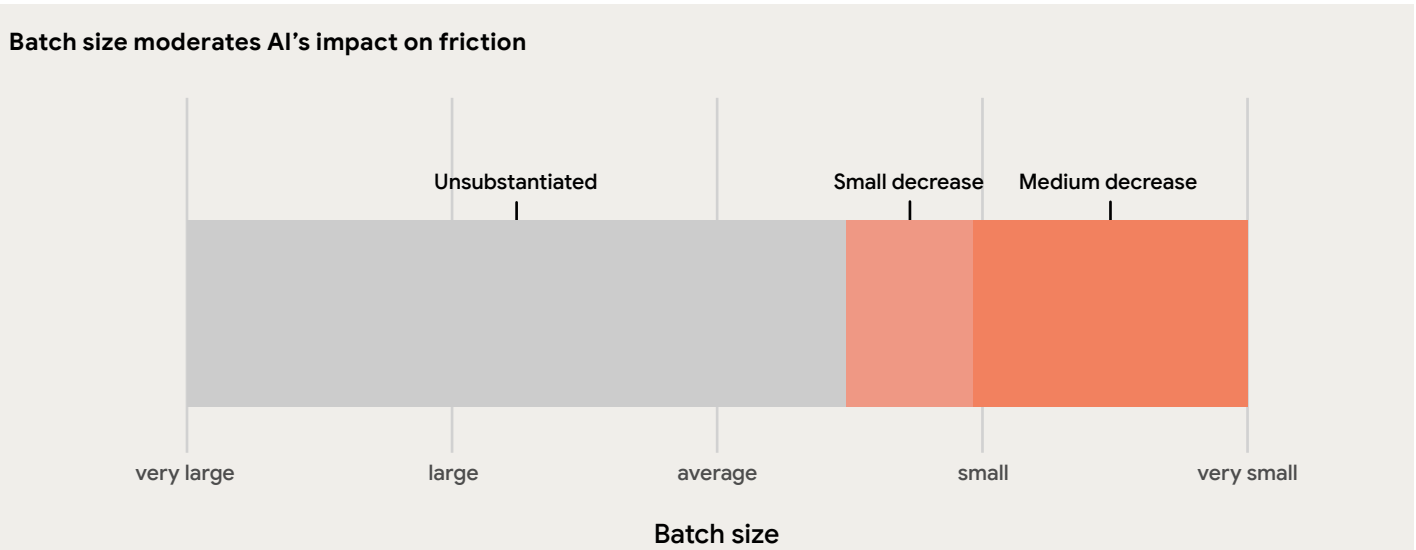


Figure 42: Batch size moderates AI's impact on friction



User-centric focus

A "user-centric focus" is also a measure that has been included in past iterations of our annual survey. A user-centric focus refers to the degree to which teams think about the experiences of the end users of their primary application or service.

The extent of a respondent's user-centric focus is measured as a single factor, comprised of three seven-point Likert scale indicators, measuring the degree to which respondents agree that:

1. creating value for users is their focus;
2. users' experience is their top priority; and
3. focusing on the user is key to the success of the business.

In this way, a team that has a user-centric focus is one that prioritizes user experience and understands its connection to business success.

With a high degree of certainty, we found that AI adoption's impacts depend on teams having a user-centric focus. Specifically, when used on teams that adopt a user-centric focus, AI's positive influence on reported team performance is amplified.

Importantly, we also found that, in the absence of a user-centric focus, AI adoption has a negative impact on team performance.

Together, these findings show that investing in developing a user-centric focus can result in important benefits for performance on AI-assisted teams—and failing to do so can be detrimental. Without a user-centric focus, AI adoption is unlikely to help teams. It may even harm them.

We have long held that a user-centric focus can help teams to clarify their goals and orient toward a shared strategy, where user experience serves as a North Star. This appears to be especially true for AI-assisted development teams; they receive an even greater benefit from AI when they center their users, and experience negative impacts from AI adoption when they do not.

These findings suggest that organizations that encourage AI adoption will benefit from incorporating a rich understanding of their end users, their goals, and their feedback into their product roadmaps and strategies. They also offer an important warning: In the absence of a user-centric focus that prioritizes meeting the needs of end users, AI adoption can hurt your team's performance.

"100% **that's why I have been here for five years—I feel that I am doing something meaningful and helping a regular person**. So, even if I'm doing it for one person, that's huge for me. But, here, I'm doing it for many millions of users ... When there are some bad days, like I'm not in the greatest mood, [if] I know whatever I'm doing is for this purpose, then that thought makes my day much much better and **motivates me to work, basically, that I'm going to help 100,000 users** this year by just developing this small feature."

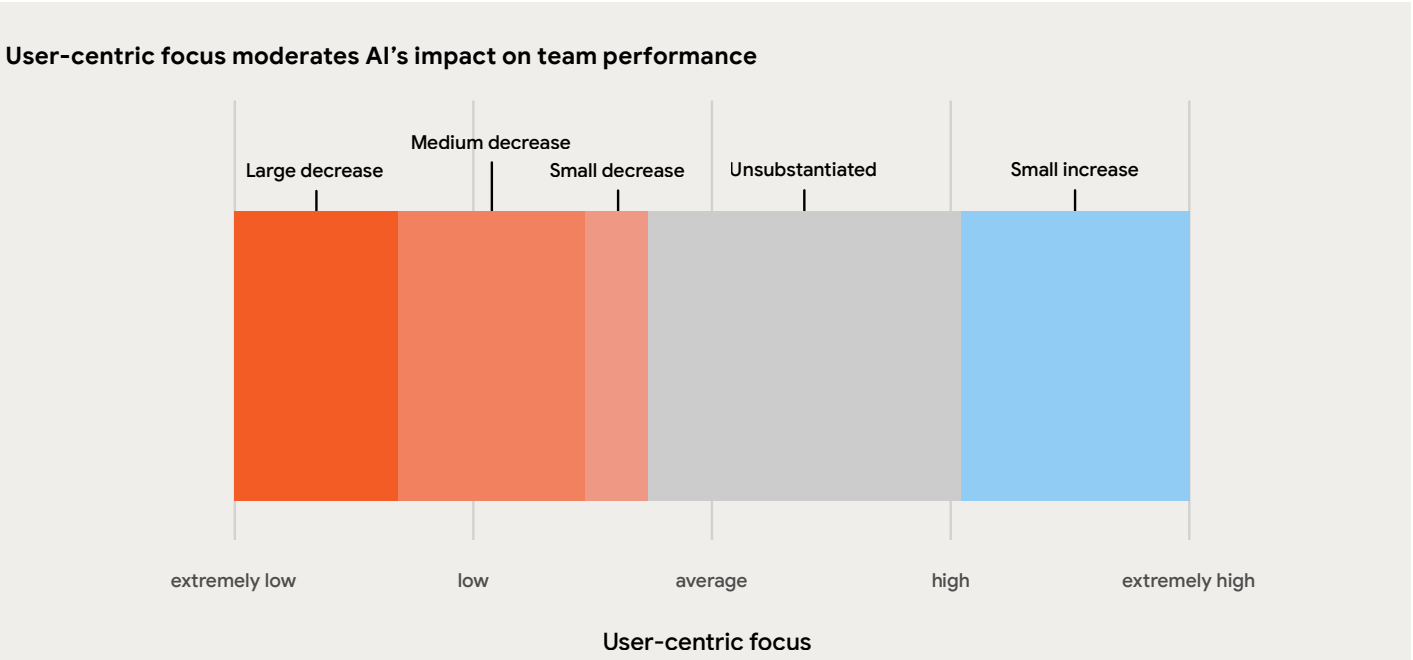


Figure 43: User-centric focus moderates AI's impact on team performance

IKS HEALTH

"A key 'a-ha' moment was realizing that clinicians don't need more tools—they need less noise. The initial assumption was that value would come from giving clinicians more advanced capabilities. What we uncovered was the opposite: simplicity and invisibility of technology were the real innovations. An AI–human hybrid model delivered superior accuracy, empathy, and trust compared to standalone automation."



Quality internal platforms

Understanding the benefits of quality internal platforms has been part of our survey in the past. In our survey, "platforms" refer to a set of capabilities that is shared across multiple applications or services, directed at making these capabilities widely available across the organization.

The quality of these platforms is measured as a single score, indicating how many of 12 characteristics a respondent indicates their internal platforms have. Please refer to the Appendix for a complete list of characteristics defining a quality internal platform in our survey.

With a high degree of certainty, we found that AI adoption's impacts depend on organizations having quality internal platforms. Specifically, in organizations with quality internal platforms, AI's positive influence on organizational performance is amplified.

Conversely, we found that AI's neutral effect on respondents' reported experiences of friction is made harmful. That is, respondents experience more friction in organizations with quality internal platforms.

Despite these mixed results, we believe that, overall, these findings point to a net-positive impact of quality internal platforms for AI-assisted teams. While quality internal platforms increase individual effectiveness by providing a uniform set of capabilities on which development teams can easily build, the standards set by internal platforms may also dictate boundaries around how development tools can be used, for instance, by defining internal-only APIs with higher security controls than their external counterparts.

In this way, quality internal platforms can serve their function both by increasing access to desired capabilities and by limiting access to undesired capabilities.

Because we have not yet arrived at a standardized set of best practices for using AI-assisted development tools, we hypothesize that quality internal platforms may predominantly have the latter effect in this space—preventing inappropriate use. This could explain increases in friction for heavy AI adopters, which may not necessarily be a negative consequence for the organization.

For this reason, and due to their benefits for organizational performance, we believe designing and maintaining quality internal development platforms is an important capability for organizations to successfully develop software in an AI-assisted environment.

Internal platforms moderate AI's impact on organization performance

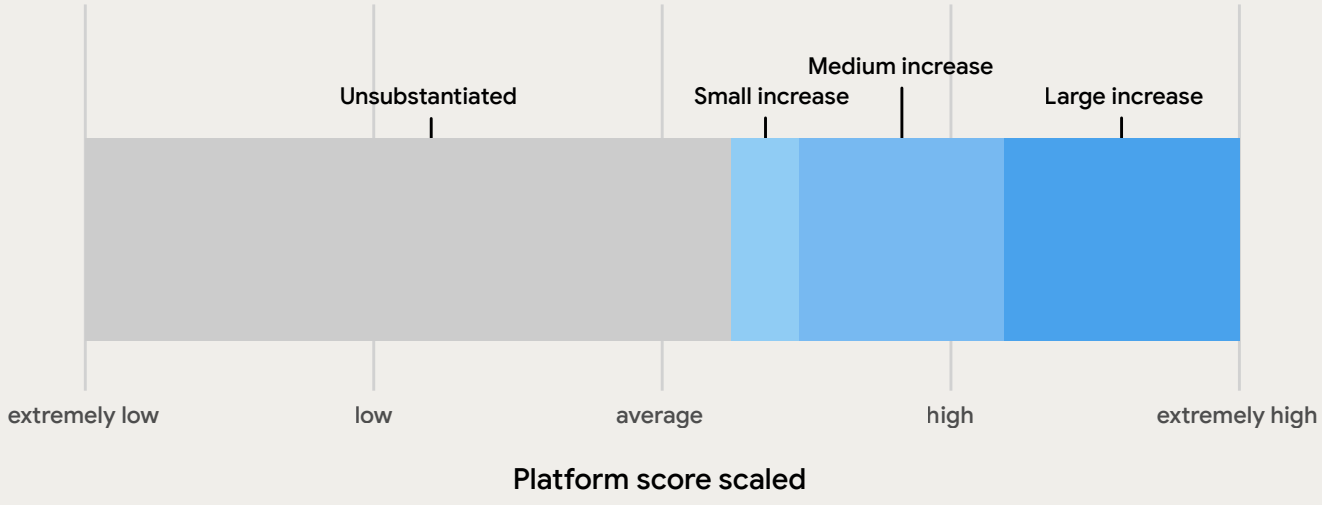


Figure 44: Internal platforms moderate AI's impact on organization performance

Putting the DORA AI Capabilities Model into practice

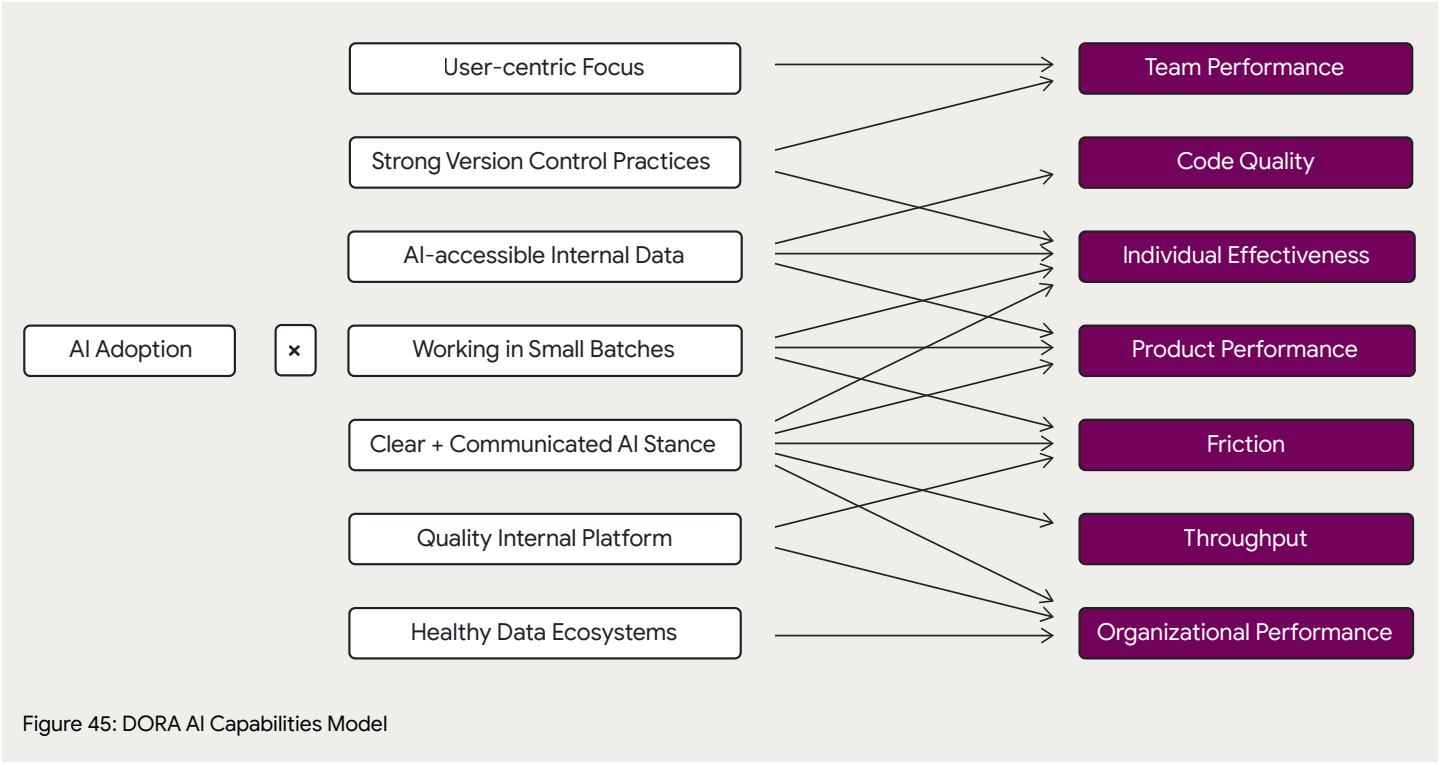


Figure 45: DORA AI Capabilities Model

The findings from this chapter suggest that successfully leveraging AI in software development is not as simple as just adopting new tools. Rather, organizations must cultivate a specific technical and cultural environment to reap the greatest rewards.



Here is some practical advice based on the seven DORA AI Capabilities:

Clarify and socialize your AI policies

Ambiguity around AI stifles adoption and creates risk. Establish and socialize a clear policy on permitted tools and usage to build developer trust. This clarity provides the psychological safety needed for effective experimentation, reducing friction and amplifying AI's positive impact on individual effectiveness and organizational performance.

Treat your data as a strategic asset

The benefits of AI on organizational performance are significantly amplified by a healthy data ecosystem. Invest in the quality, accessibility, and unification of your internal data sources. When your AI tools can learn from high-quality internal data, their value to your organization increases.

Connect AI to your internal context

Connect your AI tools to your internal systems to move beyond generic assistance and unlock boosts in individual effectiveness and code quality. This means going beyond simply procuring licenses, and investing the engineering effort to give your AI tools secure access to internal documentation, codebases, and other data sources. This provides the company-specific context necessary for the tools to be maximally effective.

Embrace and fortify your safety nets

AI-assisted coding can increase the volume and velocity of changes, which can also lead to more instability. Your version control system is a critical safety net. Encourage teams to become highly proficient in using rollback and revert features, as this practice is associated with better team performance in an AI-assisted environment.

Reduce the size of work items

While AI can increase perceptions of individual effectiveness by generating large amounts of code, our findings show this isn't necessarily the most important metric. Instead, focus on outcomes. Enforce the discipline of working in small batches, which improves product performance and reduces friction for AI-assisted teams.

Center users' needs in product strategy

Individuals can experience large increases in their personal effectiveness when they adopt AI. But, if their users' needs aren't their focus, they may be moving quickly in the wrong direction. We found that adopting AI-assisted development tools can harm teams that don't have a user-centric focus. Conversely, keeping the users' needs as a product's North Star can guide AI-assisted developers toward appropriate goals and has an exceptionally strong positive effect on the performance of teams using AI.

Invest in your internal platform

A quality internal platform is a key enabler for magnifying the positive effects of AI on organizational performance. These platforms provide the necessary guardrails and shared capabilities that allow AI benefits to scale effectively and securely across the organization.

1. "DORA's Research Program." https://dora.dev/research
2. "Fostering developers' trust in generative artificial intelligence." https://dora.dev/research/ai/trust-in-ai
3. "Concerns beyond the accuracy of AI output." https://dora.dev/research/ai/concerns-beyond-accuracy-of-ai-output
4. "Helping developers adopt generative AI: Four practical strategies for organizations." https://dora.dev/research/ai/adopt-gen-ai
5. "Working in small batches." https://dora.dev/capabilities/working-in-small-batches
6. "DORA's Research Program." https://dora.dev/research

Platform engineering

Eric Maxwell
Lead, 10x Technology, Google Cloud

Benjamin Good
Lead, Platform Engineer, Google Cloud



Our 2024 report began our exploration into the effects of internal platforms on software delivery performance.[1] We found that platforms have positive impacts on organizational performance and productivity. However, the benefits came with a trade-off: an increase in software delivery instability and a decrease in throughput.

This year's research moves beyond confirming the value of platform engineering to explore how successful platforms operate and deliver value. The data reveals that platforms are not just a collection of tools, but a holistic experience that directly impacts performance, well-being, and an organization's ability to capitalize on transformative technologies. We found that users perceive their platform as a single entity; its overall effectiveness matters more than the quality of any individual feature in the platform.

This chapter unpacks the key patterns defining the state of platform engineering from ubiquitous adoption and team structures to its crucial role as a strategic foundation for innovation and risk management.

Our key findings

Platform adoption is nearly universal: 90%.

Dedicated platform teams are the dominant organizational model, making up 76%. This shifts the leadership challenge from adoption to effective governance of a multi-team landscape.

A high-quality platform amplifies the effects of AI adoption on organizational performance. The positive impact of AI on organizational performance is strong when platform quality is high.

High-quality platforms are a force multiplier, improving organizational performance, productivity, and team well-being.

A platform should be seen as a holistic entity that enables a great developer experience.

A platform functions as an engine for managing risk, enabling speed and experimentation that corresponds to a small but credible increase in software delivery instability: a manageable tradeoff for higher performance overall.

The platform landscape: ubiquitous, complex, and team-driven

Most organizations now have an internal platform, showing the conversation has shifted from if a platform is needed to how should a platform be built. Our data shows that 90% of organizations have adopted at least one platform, with 29% of organizations now using a multi-platform environment.

Appropriately, 76% of organizations have at least one dedicated platform team, and more than a quarter of all respondents (29%) work in an organization with multiple platform teams. The prevalence of multi-platform use and multiple dedicated platform teams is less a sign of redundant tooling, and more a reflection that organizations are moving past a one-size-fits-all model. Instead they are creating federated and specialized platforms and teams to serve distinct domains and technology stacks.

The challenge for leaders has shifted from simply "having a platform" to "governing a complex platform of platforms." Like applications, platforms will benefit from adopting the DORA capability of loosely coupled teams,[2] to help manage complexity. Doing so requires establishing clear charters and interfaces between teams to ensure their ecosystem collectively improves developer experience rather than creating new organizational silos.

Overall experience is what matters

We asked respondents to rate their platforms based on how well their platform(s) perform certain capabilities. For example, "The platform helps me build and run secure applications and services."

We found an experience gap. Core technical capabilities, such as, "the platform aids with reliability and security," are perceived as well-provided, while user experience features, including "acting on feedback" and "how well tasks are automated," lag slightly behind.

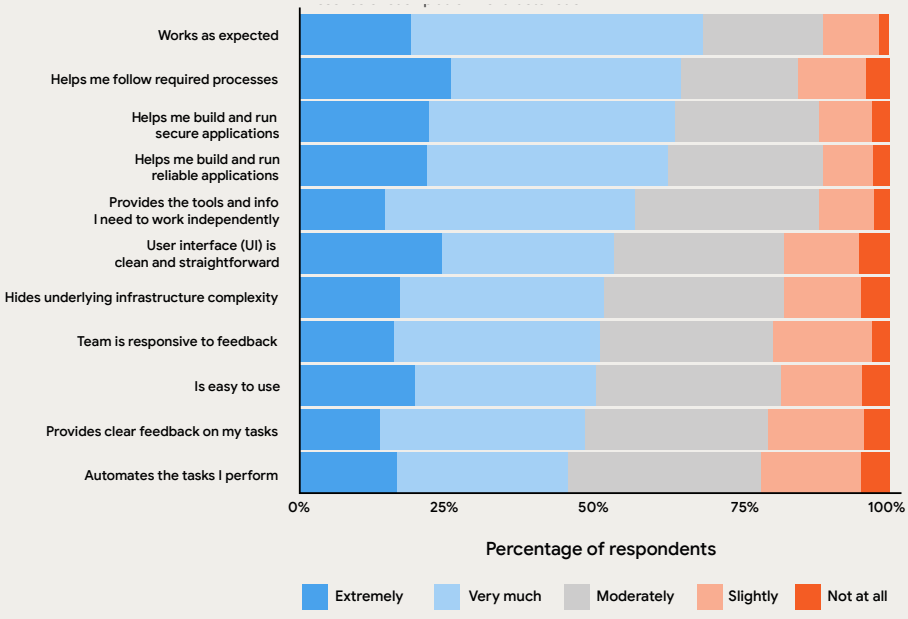


Figure 46: Perceptions of platform capabilities

A developer's overall impression of the platform, as a helpful partner or as a source of friction, heavily colors their rating of every specific feature. The relatively tight grouping of capabilities shows that respondents don't perceive the platform as a checklist of discrete parts; they experience it as a single entity.

The experience gap likely reflects platforms where technical foundations are built first.

This data shows that until the user experience is addressed, the platform's full value remains unrealized.

Embracing the mindset of the platform as a product, meaning you think of your internal development tooling as products and your developers as your customers, helps ensure the user remains the focus—a key finding in the 2024 report.

User-centricity is how you avoid some of the common pitfalls when building internal developer platforms, such as:

- **Build it and they will come:** A team builds a platform based on what they think developers need, without doing any user research, interviews, or validation. They focus entirely on the technology and engineering, assuming its value will be self-evident.
- **Why it fails:** The platform ends up being a ghost town because it doesn't solve real, painful problems for developers or it doesn't fit their existing workflows.
- A product approach starts with developer empathy and discovery. The platform team continuously engages with its users to understand their biggest challenges, ensuring they build something people actually want and will use.
- **The ticket-ops trap:** The platform team operates like a vending machine for infrastructure. They don't have a vision or a roadmap; their work is entirely reactive and driven by an endless queue of tickets from developers (such as, "Provision me a database," "Set up a CI/CD pipeline.")
- **Why it fails:** This creates a bottleneck and adds toil for both the platform team and the developers. The team spends all its time on one-off requests, and never has capacity to build cohesive, self-service capabilities.
- A product approach focuses on building a self-service platform with a clear roadmap. The goal is to eliminate ticket queues by empowering developers to provision resources themselves through automated, reusable tools and golden paths.
- **The ivory tower platform:** Here, a central team dictates the platform's architecture and tools from on high, enforcing rigid standards without collaboration or a feedback loop. They act as gatekeepers of technology rather than enablers of developers.
- **Why it fails:** This approach can leave developers feeling disempowered and often create shadow IT or unofficial workarounds to bypass the platform's constraints, defeating its purpose.
- A product manager for the platform actively seeks feedback and treats developers as customers. The platform is designed to be enabling, not just restrictive, offering paved roads that are easy and desirable to use, but not necessarily mandatory.

Correlation matrix of platform capabilities

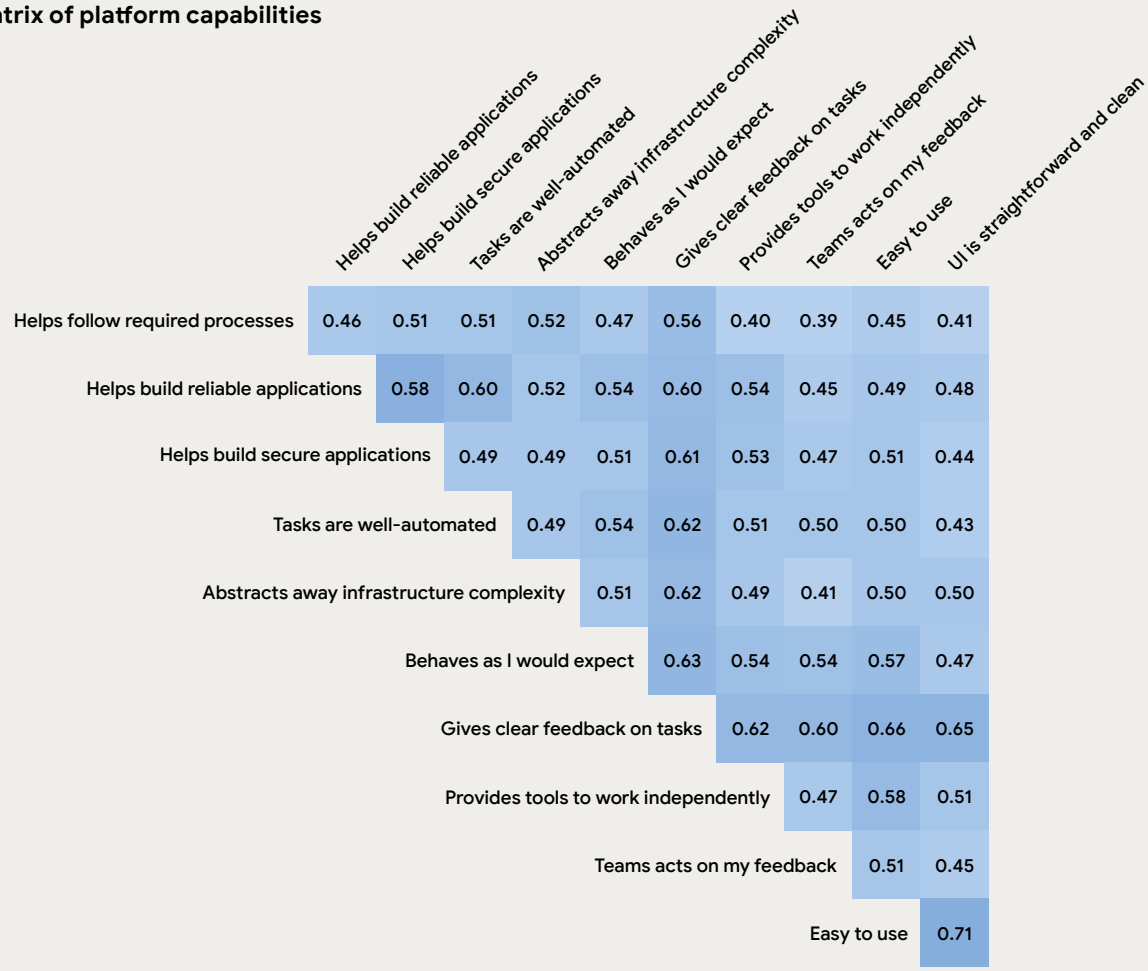


Figure 47: Correlation matrix of platform capabilities

When considering the capabilities and how they relate to each other, all capabilities are more or less evenly correlated. However, there are two capabilities that stand out from the others.

First, the capability most correlated with a positive user experience is providing clear feedback to tasks.

When the platform provides clear feedback when a task succeeds or fails, users feel empowered to act, troubleshoot, and take the next action, instead of having to sift through things and figure it out on their own.

Second, the "UI is straightforward and clean" has a lower correlation: while having a clean UI might improve perceptions, it doesn't necessarily translate to having an effective platform.

Even though providing clear feedback and UI are tightly correlated to other capabilities, they are still rated among the lowest in terms of perception. What this means is that focusing on improving a single capability in isolation is a flawed strategy.

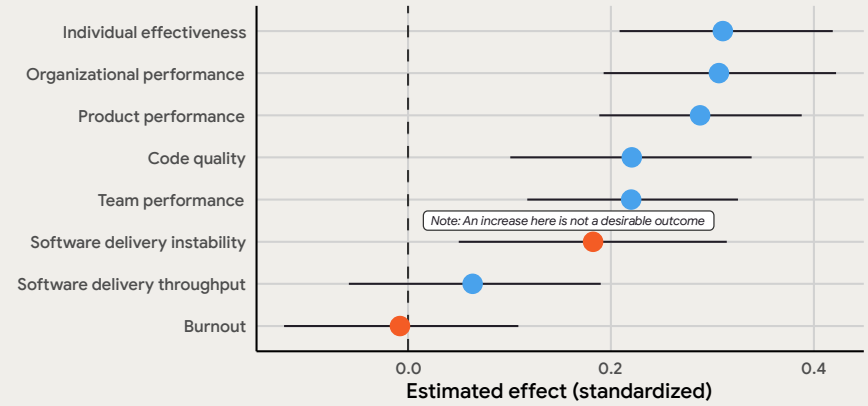
To improve the perceived quality of the platform, teams must treat it as a holistic internal product and focus on improving the entire developer journey. If a platform is technically excellent but not user-centric (see the 2024 DORA report),[3] it cannot be considered a success.

A force multiplier for performance, well-being, and risk

A high-quality platform, as defined by our capabilities, has a broad, statistically positive impact across the board. It's linked to higher organizational performance, product performance, and productivity.

Consistent with past research, we found that a better platform is associated with a small but credible increase in software delivery instability, meaning a higher change failure rate and increased rework.

Estimated effect of quality internal platforms on key outcomes



For outcomes in orange, such as Burnout, a negative effect is desirable.
89% credible intervals
Figure 48: Estimated effect of quality internal platforms on key outcomes

"In general, the idea was to try to do continuous delivery as much as possible, if you get it into the main line, ship it out, try to have some tests in, you know, to try to catch problems, but **just deploy it. And for the most part, that was fine because even if something failed during deployment, there were a lot of little things to keep the site from going down.** So, a deployment might fail, and we might need to go quickly fix that. But, the site's still running."

A great platform acts as a force multiplier, which translates directly into better performance and productivity. The corresponding increase in instability may be representative of a healthy high-velocity system, where the additional instability is acceptable as long as it doesn't impact product performance.

The increase in performance in spite of the increase in stability, suggests a form of risk compensation where the platform makes it fast and cheap to recover from failure and teams are empowered to experiment more and accept a higher rate of minor failures in pursuit of speed.

The slight increase in instability should be seen as a manageable trade-off for the significant gains in performance that the platform enables. The improvements in product performance are likely more impactful than the modest gains in delivery throughput and reduction in delivery stability.

Investing in a high-quality platform is a powerful strategic lever with widespread returns.

The strategic imperative: Your platform is the key to unlocking AI

The positive effect of AI adoption on organizational performance depends on the quality of the internal platform. AI adoption has a negligible effect on organizational performance when platform quality is low, but when platform quality is high, the effect is strong and positive. This is a critical finding for any leader investing in AI.

Internal platforms moderate AI's impact on organization performance

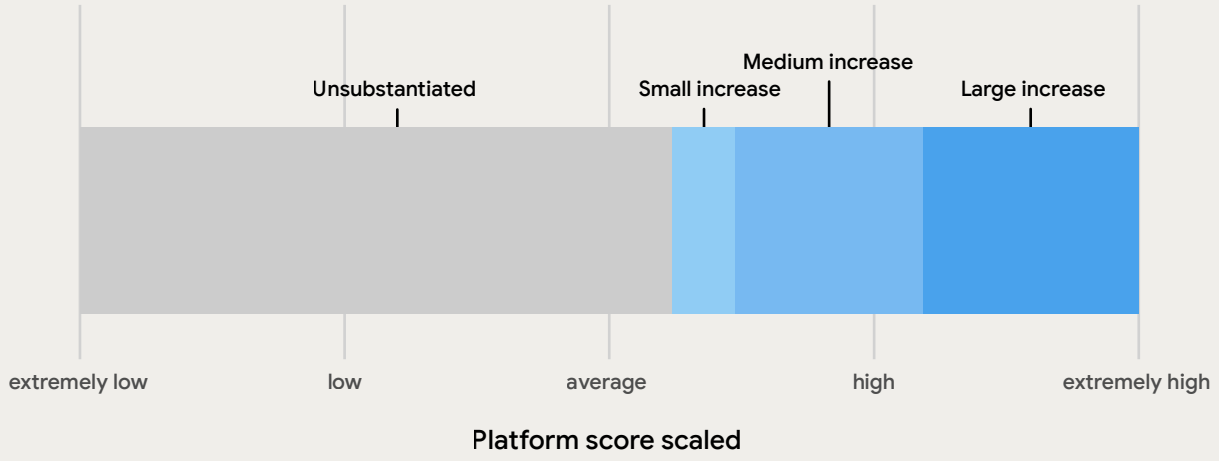


Figure 49: Internal platforms moderate AI's impact on organization performance

wayfair

"Wayfair learned that the biggest gains came not just from detecting failures faster, but from reducing the effort required to fix them. Embedding AI into the CI/CD loop showed that developers engage most with targeted, explainable suggestions and auto-generated fixes when these are delivered seamlessly in the tools they already use."

A high-quality platform serves two purposes when amplifying the impacts of AI on organizational performance. First, it acts as the distribution and governance layer required to scale the benefits of AI from individual productivity gains to systemic organizational improvements.

Without this foundation, AI adoption remains a series of disconnected local optimizations. The platform provides the centralized context and abstracts away the difficult or tedious parts of making AI usable and effective at scale. That said, AI is changing at a rapid pace. Be cautious not to overly standardize AI practices, tools and methods, as that will likely limit the positive impacts and ability to adapt as AI changes.

Second, as the platform serves as a risk mitigator when not considering AI, its risk mitigation effects are similarly useful for AI. The platform should be used to create a safe space, allowing individuals to learn and experiment. The safe space for experimentation will help the platform and platform teams to grow and adapt to better support new models, interaction modes and ways of developing applications.

Additionally, whether code is created by hand or by AI, the same automated testing and deployment processes are applied, essentially helping to make sure the changes introduced into applications and services are safe.

An investment in AI without a corresponding investment in high-quality platforms is unlikely to yield significant returns at the organizational level. To truly leverage AI for competitive advantage, leaders must view platform engineering as a foundational strategic enabler.

Three imperatives for the platform era

Embrace the holistic experience

You can't fix a bad platform by improving one feature.

Treat your platform as a whole product, focusing on the entire developer journey from feedback loops to automation.

Make your platform the foundation for AI

Your platform is the strategic prerequisite for unlocking the organizational value of AI.

It is the engine that will turn your AI investments into a true competitive advantage.

Use your platform to calibrate your risk appetite

A great platform changes your organization's relationship with risk by making failure cheap and easily reversed.

Understand and manage the trade-off between the velocity this enables and the resulting instability, recognizing that the platform does not eliminate the local impacts of that risk.

1. Accelerate State of DevOps 2024. https://dora.dev/dora-report-2024
2. "Loosely coupled teams." https://dora.dev/capabilities/loosely-coupled-teams
3. Accelerate State of DevOps 2024. https://dora.dev/dora-report-2024

Value stream management

Rob Edwards
Application Delivery Lead, Google Cloud



Every organization is under pressure to innovate faster. We're all adopting AI, automating processes, building platforms, and shipping features at a breakneck pace. But are we actually getting better? Or are we just getting faster at creating features that don't deliver value, faster at burning out our teams, and faster at introducing complexity? The greatest risk today isn't falling behind, it's pouring massive investment into chaotic activity that doesn't move the needle.

For over a decade, DORA's research has been guided by a core belief: The highest-performing organizations don't just adopt new tools, they become experts in the system of delivering value. They have a proven capability for "getting better at getting better." They can understand their workflow, identify their true constraints, and apply their resources with intention and focus.

This year, we've confirmed that the capability to elaborate on and manage your value stream is what truly separates disorganized activity from focused improvement. In our 2025 research, we found that teams who focus on understanding their value streams dedicate significantly more of their time to valuable work.

More importantly, we've discovered that value stream management (VSM) is the force multiplier that turns AI investment into a competitive advantage, ensuring that this powerful new technology solves the right problems instead of just creating more chaos.

How to achieve focused improvement: The principles of value stream management

Value stream management (VSM) is the practice of visualizing, analyzing, and improving the flow of work from idea to customer. It is not a heavyweight process, but a set of four principles for achieving the clarity needed to focus on improvements where they count most.

For a detailed, step-by-step guide on how to conduct a value stream mapping exercise, see the DORA Value Stream Management guide.[1]



From mental mess to shared map

Trying to get your head around a complex system is tough. It's a huge mental drain for anyone to remember all the intricate details, which gets in the way of understanding the bigger picture.

When a team collectively maps out a system, it gets all those details out of their heads and into a shared space. Suddenly, the system's structure—and any hidden patterns—become obvious. This visibility makes it much easier to have a real conversation about what's working and what isn't. At its core, this is exactly what VSM is all about.

The practice itself is about charting the entire software delivery lifecycle, from initial concept all the way to the customer.

This map covers everything: product discovery, design, development, testing, deployment, and operations. Creating this shared representation allows teams to develop a collective understanding of the workflow, making it much easier to spot the real bottlenecks and inefficiencies that hold things up.

Mapping the entire system from concept to customer is the goal, but you don't have to tackle it all at once. The key is to start where you can have the most impact.

Before diving in, take a high-level look at your workflow to identify the primary constraint so that you don't optimize part of the process that isn't the real bottleneck. If your team's biggest challenges lie in product discovery, for instance, that may be a more effective place to begin.

Still, a powerful and proven starting point, one DORA has used for years, is the scope from code commit to production. We start there because this part of the process can be most readily standardized and tuned for efficiency.

More importantly, it's the phase where teams typically have the most agency, so they can make immediate, impactful improvements. It stands in contrast to discovery work, where the primary goal is optimizing for effectiveness.

Successfully completing this core process creates quick wins, building the momentum and credibility needed to influence the broader system of product discovery and customer feedback.

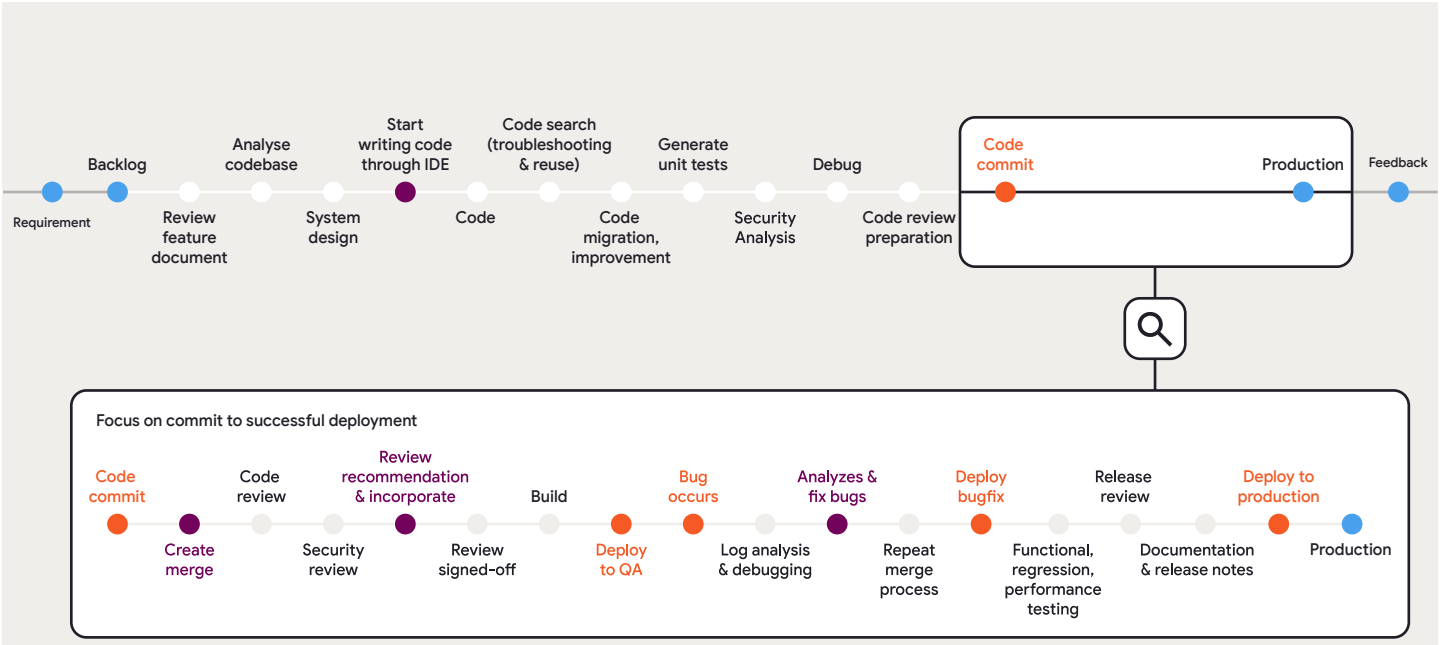


Figure 50: An example value stream map showing the flow of a backlog item to production

Focus on flow, not just speed

Once you've mapped your value stream, the real goal is to get work flowing smoothly and predictably. Doing so requires a shift from focusing on local efficiencies to optimizing the system as a whole.

You should start by measuring what matters. Track key metrics like lead time, process time, and the ratio of value-add to wait time. These numbers give you data on your true constraints and provide a clear baseline, so you'll know if your improvements are actually working.

This systems-level view is crucial for identifying the best places to apply new solutions or technologies. For example, a team may discover, through mapping, that code reviews are a significant bottleneck. With this insight, they can decide to apply AI to improve the code review process, rather than using AI to simply generate more code that will only exacerbate the bottleneck.

The real win is using AI to improve the code review process itself, clearing the actual blockage in the system. That's what focusing on flow is all about: You want to solve the whole system's biggest problem, not just speed up a single step.

Create a culture of continuous improvement

VSM is not a one-time exercise, but an ongoing cycle of improvement. Revisit the value stream map your team created regularly and treat it as the starting point for every improvement discussion.

It's essential to foster a culture where teams are empowered to experiment, learn, and adapt.[2] This means stating clear goals, but also providing teams with the autonomy to figure out how to achieve them.

They need the freedom to experiment, learn, and adapt without a fear of reprisals. It's also crucial to encourage teams to share what they learn with the rest of the organization.

By approaching VSM this way and keeping a record of each iterative improvement, you create a living history of your progress. Over time, you can reflect and establish a clear story of how each change sharpened your ability to deliver value to customers.

> "Then there's, kind of, the complexity of it like, 'How much work is it going to take?' and like, 'How long will it end up taking?' 'How do you estimate that?' It's always uncertain and then there's, like, a lack of control. Are we going to be able to make changes that we need on the fly and get things done?
>
> Or, are we going to have to go through a two- or three-week process every time we want, like, a firewall rule change? Something like that. There's just so many different ways that [the process] would have dragged down our work. I think [especially] the more senior people were very familiar with that from working in other big companies."

Build on a foundation of technical excellence

A fast, smooth flow is impossible without a foundation of technical excellence, and that foundation is typically a well-designed internal platform. By giving developers "paved roads" for capabilities like testing and delivery, a strong platform abstracts complexity and makes high-performance work scalable.

The relationship between a strong technical foundation and organizational performance is explored in detail in the Platform Engineering chapter.

How this appears in our 2025 findings

For years, DORA has advocated for using practices like VSM to create a fast flow of work.[3,4] But does that advice still hold up, especially with the widespread adoption of AI?

This year, we wanted to validate our long-standing hypotheses about the benefits of VSM. Our findings confirm that organizations that embrace the principles of VSM see significant, measurable benefits.

First, our research confirms that VSM practices have a direct and powerful impact on performance. We found strong evidence for the following:

- **VSM drives team performance.** Teams that consistently review and improve their value stream report markedly higher performance.
- **VSM leads to more valuable work.** These teams spend significantly more of their time on work that matters to the organization and its customers.
- **VSM improves product performance.** Ultimately, this focus on the value stream translates into better product outcomes, which is arguably the most important result.

This data tells a simple, human story: Teams that work together to understand their value stream spend more time on work that matters. When teams share a clear understanding of their entire value stream, they can focus their efforts on what matters most, translating that clarity into meaningful impact.

This clarity is the key to unlocking new technologies like AI. Instead of just throwing tools at a problem, it allows teams to be strategic. You're no longer just optimizing a small step; you're removing friction from the system's biggest constraint.

This belief led us to formulate a key hypothesis for this year's research:

- **VSM turns AI into an organizational advantage:** We hypothesize that VSM moderates the relationship between AI adoption and organizational performance. Teams with mature VSM practices can channel the productivity gains from AI toward solving system-level problems, ensuring that individual improvements translate into broader organizational success. Without VSM, AI risks creating localized efficiencies that are simply absorbed by downstream bottlenecks, delivering no real value to the organization as a whole.



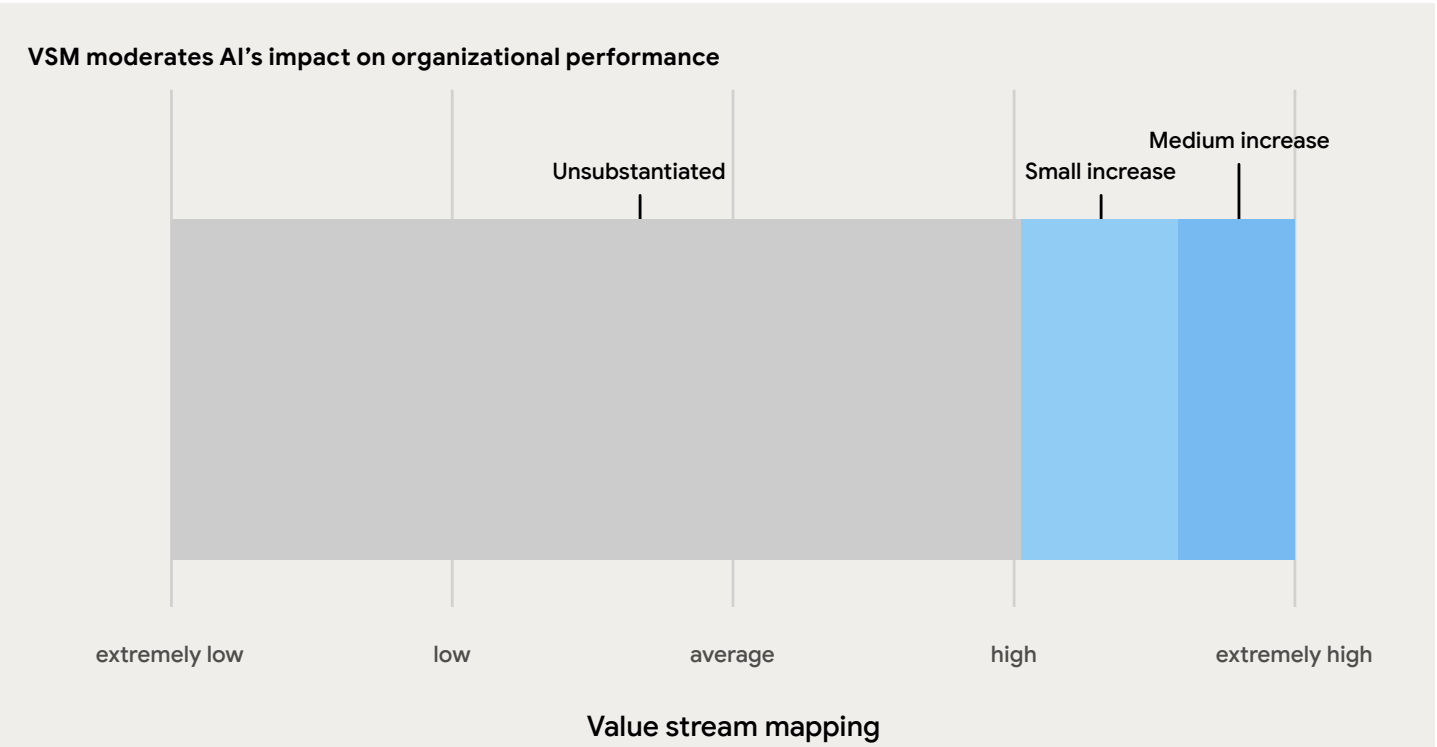


Figure 51: VSM moderates AI's impact on organizational performance

Conclusion

Our analysis validates this hypothesis. While AI adoption on its own shows a modest impact, the effect is dramatically amplified in organizations with strong VSM practices.

This confirms that VSM is a critical enabler for getting the most out of your AI investments. By ensuring that team-level and individual productivity gains are focused on the most important system-level constraints, VSM helps translate local improvements into meaningful organizational impact.

This research is not just an observation; it is a challenge. The first step to escaping the cycle of chaotic activity is simple, but not easy: Ask your team, "Can we draw our software delivery value stream on a whiteboard?"

If the answer is no, or if the drawing reveals more questions than answers, you have found your starting point. That single conversation is the beginning of getting better at getting better.

1. "How to use value stream mapping to improve software delivery: A guide to value stream mapping." https://dora.dev/guides/value-stream-management
2. "How to transform your organization." https://dora.dev/guides/how-to-transform
3. "Visibility of work in the value stream." https://dora.dev/capabilities/work-visibility-in-value-stream
4. "Visual Management." https://dora.dev/capabilities/visual-management

The AI mirror: How AI reflects and amplifies your organization's true capabilities

Eirini Kalliamvakou, Ph.D.
Office of the CEO Research Advisor, GitHub



Last year's DORA report found that teams using AI reported lower throughput and more instability in their software delivery. The unexpected findings sparked a lively debate—how could a technology built to accelerate work be linked to slower, shakier outcomes?

This year, the picture has shifted a bit. AI adoption has now helped throughput tick upward, but instability still lingers. That mix of progress and friction has led us to dig deeper and look beyond simple binary AI comparisons to understand what really determines its impact.

What we found points to something bigger than tools and skills: More than anything, the environment that AI is nested in shapes its impact.

Looking beyond the tools to drive AI impact

One of the most exciting outcomes from this year's research is the DORA AI capabilities model. The effect of AI use on outcomes like throughput, code quality, and team and organizational performance was consistently amplified by seven capabilities:

1. A clear and communicated AI stance
2. A healthy data ecosystem
3. AI-accessible internal data
4. Strong version control practices
5. Working in small batches
6. User-centric focus
7. A quality internal platform

These systemic conditions reflect how an organization structures its work, supports its teams, and aligns its environment to modern development practices. These capabilities can help determine whether using AI tools translates into meaningful results, and that makes them amplifiers. The fact that they are all team- and organization-level reinforces a critical shift we need to make in how we think about AI's role in software delivery.

We are seeing that AI's effects on performance depend on the system in which the work takes place. For example, a healthy data ecosystem integrated with AI tools can help create conditions that can boost AI benefits from individual productivity improvements to organizational leaps. Without those foundational efforts to set up AI users for success, its benefits may stall, plateau, or stay unevenly distributed.

To take advantage of this insight, we should move the spotlight from how individuals use AI to how organizations design the systems around them.

Organizations are systems, not sums of individuals

To understand what is needed to scale AI impact from individual productivity gains to organizational-level benefits, we need to think about systems. Organizations are less like collections of individuals and tools, and more like networks of interdependent parts. Work flows through teams, processes, policies, infrastructure, and shared norms. While individual capabilities play a crucial role in shaping outcomes, overall performance emerges from how all these parts interact.

This idea is central to systems thinking, a perspective that has shaped how high-performing organizations evolve. W. Edwards Deming,[1] one of the founding figures of modern quality management, argued that most performance issues stem not from people, but from systems. As he famously put it, "A bad system will beat a good person every time."

In a system, improving one part does not guarantee better outcomes overall. In fact, local improvements can be blocked, diluted, or even reversed if the rest of the system isn't able to adapt.

This is also the core insight of the Theory of Constraints.[2] Every system has a limiting factor—a constraint—that governs how much value it can deliver. Focusing on anything other than the constraint might feel productive, but it won't meaningfully improve value flow.

This has implications for AI adoption. When developers use AI tools and write code faster, the code still needs to go through testing and review queues, followed by integration and deployment processes.

The overall pace of delivery is unlikely to change significantly unless the surrounding workflows are updated for developers' new tools and increased speed. The system isn't designed to carry the gains, let alone amplify them.

We have seen this story before. During the shift to cloud, companies that simply moved infrastructure without rethinking architecture and delivery practices saw limited returns.

However, organizations that restructured their applications, teams, and operations for cloud-native workflows were able to unlock real value. The same was true for Agile and DevOps: Both delivered on their promises only when they were paired with deep changes to roles, feedback loops, and team boundaries.

New, powerful technologies and tools produce corresponding transformative results only when the system around them evolves.

That's why AI adoption needs to be treated as a transformation effort. If the organization wants to move faster, experiment more, and shift how developers spend their time, it will need to revisit how work itself flows.

Are downstream systems—like integration, testing, deployment, and compliance—flexible and responsive enough to harness AI's speed? Are decision-making structures keeping up with the tempo of work? Are teams incentivized to delegate tasks, verify AI output at scale, and share knowledge in new ways?

Without intentional changes to workflows, roles, governance, and cultural expectations, AI tools are likely to remain isolated boosts in an otherwise unchanged system—a missed opportunity. To scale AI's impact, organizations should invest in redesigning their systems. That means identifying constraints, streamlining flow, and enabling the conditions where local acceleration becomes organizational momentum.

What might that transformation look like?

Transformation in two ways: Augmenting and evolving

As organizations pursue the full value of AI, we can think about transformation along two complementary paths. One focuses on augmenting existing systems to remove friction and support the velocity introduced by AI tools. The other imagines how AI opens the door to new ways of working altogether.

Augmenting: Preparing systems to carry the gains

When developers begin using AI tools and experience productivity improvements, but teams don't see a corresponding boost in throughput or delivery speed, the system itself may be the limiting factor.

Augmentation means identifying and resolving those friction points so that individual acceleration can flow downstream.

Code reviews and handoffs: Consider where AI can accelerate and clarify existing steps. For example, AI-generated first-pass reviews can reveal issues quickly and reduce time spent on routine feedback. Structuring AI input to highlight risks or summarize diffs can also make reviews easier and faster for humans.

Integration and deployment pipelines: AI-generated code moves fast—can your systems keep up? Continuous integration and deployment pipelines may need to evolve to reduce wait states and allow for higher-frequency delivery. Quality checks powered by AI can be layered in without adding manual gates, improving flow without sacrificing assurance.

Data infrastructure: The AI capabilities model points to the value of investing in updating data infrastructure. First, AI's benefits for productivity and code quality are amplified when AI models and tools are connected to internal data.[3] Think repos, work tracking tools, documentation, and even decision logs and communication tools. Adding this valuable context improves the output of AI tools. Related to how critical context is, the second finding was that the benefits of AI for organizational performance are larger when the data ecosystem is healthy, that is, when data is AI-accessible, accurate, and complete.[4]

Security and privacy protocols: With AI now participating in development and operations workflows, security practices must evolve. This includes ensuring secure tool usage, updating policies, and introducing AI-aware monitoring systems that maintain trust without introducing bottlenecks. Automating parts of these processes can help teams maintain speed.

Change management and cultural alignment: Like any organizational transformation, AI adoption needs vision, support, and communication—all traits of transformational leadership.[5] Leaders should articulate the long-term goals of AI transformation—whether innovation, velocity, or quality—and support the transition with training, shared practices, and realistic expectations.

Culture matters, too. Teams need permission to experiment, make mistakes, build fluency, and share what they learn. Rewarding behaviors like verification, delegation, prompt curation, and agent orchestration sends the right signals about what success looks like in an AI-accelerated environment.

Evolving: Designing for what AI makes possible

Beyond augmenting existing systems, AI offers a chance to design new workflows—approaches that are native to how AI operates.

AI-native delivery pipelines: AI can continuously analyze code for bugs, security vulnerabilities, and violations of team standards. It can suggest tests and even generate them dynamically. With the right data and integrations, AI can also forecast deployment risks and performance regressions before they occur.

AI-native data systems: AI can help maintain its own environment by organizing, tagging, cleaning, and analyzing data. This enables more robust insight generation and faster iteration on data-informed decisions. It also surfaces patterns in how teams work, offering new levers for operational improvement.

AI-native security: AI can expand the capacity of security teams by detecting threats earlier, identifying anomalous behavior, and even automating parts of the incident response process. For security teams that are often under-resourced, AI's role can be to ease pressure while improving response times.

AI-native collaboration models: Emerging practices like agentic workflows and swarming are beginning to reshape how humans and AI work together. Agentic workflows assign tasks to autonomous AI agents, while swarming enables teams and AI to converge dynamically on complex problems. Though still early, these patterns hint at new, more adaptive modes of collaboration.

Emerging trends like Continuous AI[6] illustrate how AI-native workflows can be sustained over the long term. Continuous AI treats the AI system as a living part of the development pipeline and a participant in team processes.

Continuous AI perceives events happening in the project context on an ongoing basis and, by operating autonomously yet collaboratively, facilitates team interactions and adjusts direction along with the team. The key is for the AI system or agents to be constantly updated with relevant context and constantly measured for accuracy, usefulness, and cost. This keeps AI aligned with the organization's evolving architecture, practices, and priorities, ensuring that its outputs remain relevant and high-quality as the environment changes.

In both augmentation and evolution, the common thread is intention. Deploying AI tools alone will not produce transformation, but paired with both pragmatic and visionary system-level changes, adopting AI can be a catalyst for reshaping how software is built, delivered, and secured.



Where to begin: Practical steps for AI transformation

The earlier an organization begins treating AI adoption as a transformation effort, the more control it will have over how that transformation unfolds. The technology is evolving quickly, but the real differentiator lies in how effectively organizations respond. Starting before old processes solidify around new tools means organizations can shape the future of their systems, rather than inherit them by default.

A natural first step is to examine how work actually flows today. In practice, that means creating shared visibility into how ideas move from inception to delivery. This process, commonly known as value stream management, can help teams visualize each stage of delivery, from coding and code review to testing, deployment, and production.

When done well, mapping the flow of work exposes where coordination costs accumulate, where delays and rework are common, and where the system absorbs or stalls acceleration introduced by AI tools. It can help zoom in on the factors that the Theory of Constraints says will enable value the most.



To get started mapping the flow of work, organizations can form small, cross-functional working groups composed of practitioners embedded in daily software delivery operations, including engineers, product managers, data engineers, operations, and security. These groups are best positioned to map the system from the inside and surface coordination breakdowns and bottlenecks, as well as identify where AI could play a transformative role.

These efforts are most effective when they carry executive sponsorship. That endorsement signals strategic importance, ensures the work is resourced, and creates a clear path from discovery to action. The mandate of these working groups is to make strategic recommendations: Where should the system adapt to accommodate AI? Where can AI augment processes or roles? What capabilities need to be developed to unlock long-term value?

In some cases, an external facilitator or advisor can help guide the process, offer benchmarks, and keep the conversation focused on systemic opportunities that can unlock broader improvements.

For change to take root, insight must come from those inside the system. When practitioners drive the discovery and leadership commits to enabling the outcomes, the conditions for meaningful transformation start to take shape.

Critically, this work must be approached with systems thinking. As discussed earlier, organizations are complex systems and improving one node, like accelerating code generation, will not improve performance unless the adjacent components evolve in parallel. The DORA AI Capabilities Model provides insight into what organization-level interventions will amplify AI benefits.

For example, a working group might discover that while AI tools are capable of providing valuable suggestions, they frequently generate responses that miss critical context, such as team conventions, architectural history, or past incidents.

That's probably unsurprising for many organizations, since such information is often buried in disparate systems and informal knowledge channels.

In response, the group might recommend exposing internal documentation, decision records, and historical tickets to AI models in a structured, secure way. They could propose building workflows that automatically tag and surface

this context during development or review, reducing the time developers spend searching and improving the quality of AI output.

Alternatively, the working group might explore how AI could be used to identify outdated documentation, summarize long project discussions, and detect inconsistencies between what systems are doing and what documentation claims they do.

Doing so can help turn fragmented knowledge into a structured, actionable asset.

Beyond process changes, these working groups can also surface the need for new skills and roles. As developers delegate more work to AI tools, tasks like verification, orchestration, and workflow design become more central.

Organizations will need to define what those roles look like, how to support them, and how to align incentives accordingly.

This includes providing targeted training beyond tools on how AI can change the nature of development work.

None of these changes happen automatically, or all at once. They require intention, ownership, and sustained support. However, what they don't require is perfection from the outset.

When organizations start small, invest with purpose, and create shared accountability across roles, they can build momentum. Step by step, capability by capability, transformation takes root.



TBC BANK

"Cultural transformation is just as critical as tooling. Success required not only introducing automation and metrics, but also aligning teams around shared goals and ownership."

AI as mirror and multiplier

AI has the potential to reshape how software is built, but it does not change organizational systems on its own. What it does do, often very quickly, is reflect how those systems actually operate.

In well-aligned organizations, AI amplifies flow. In fragmented ones, it exposes pain points. Teams that have strong practices, flexible workflows, and shared context often see immediate benefits.

Teams that rely on brittle processes and implicit knowledge may find those gaps more visible than ever.

This is why AI functions both as a mirror and a multiplier. It shines a light on what's working, accelerating what's already in motion, but it also surfaces what needs to change.

For organizations ready to look, the reflection AI offers becomes a roadmap. As we have seen with other transformations in the past, organizations willing to treat AI adoption as an opportunity to evolve how work gets done will be the ones that benefit most—both from the tools, and from the transformation they make possible.



1. The W. Edwards Deming Institute. https://deming.org
2. "Theory of constraints." https://en.wikipedia.org/wiki/Theory_of_constraints
3. DORA AI Capabilities Model
4. Ibid.
5. "Transformational leadership." https://dora.dev/capabilities/transformational-leadership
6. "Continuous AI." https://githubnext.com/projects/continuous-ai

AI: A skill development threat—and opportunity

Matt Beane, Ph.D.

Associate Professor at the University of California, Santa Barbara, Digital Fellow at Stanford and MIT, and CEO/Cofounder of SkillBench

When it comes to skill development, software is similar to other occupations. Expertise should flow from senior staff to juniors, and ideally fresh perspectives and skills should bubble up.

Senior developers do more than review pull requests; they teach juniors how to think architecturally. Pair programming isn't just about catching bugs; it's about transmitting tacit knowledge that's hard to document. At its best, the three-generation model—junior, mid-level, senior—helps developers gain skills from joint problem-solving, not formal training.

We need to investigate the effects of AI deployment on this taken-for-granted process. I have spent my career studying intelligent automation and skill, and have shown across more than 31 occupations that default use of intelligent automation changes traditional apprenticeship models, leaving fewer opportunities for novices to engage in the hands-on work essential for their development.

Experts can self-serve, so they do. A few juniors manage to learn despite this participation barrier, but most struggle. I've studied nothing but AI use in complex task performance since 2023, most recently in software engineering, and this pattern is likewise evident in my early findings.

But there are interesting exceptions, and far more understanding is needed. Like other groundbreaking technologies in the past, from the printing press and personal computer to the Internet, AI is being developed and deployed at unprecedented speed. And we don't know how human capabilities will adapt to these changes.

Instead, many are focused on measuring AI-related productivity. We track adoption rates, lines of code generated, pull requests merged. Not metrics indicative of skill development like linguistic or stylistic diversity over time.

The best organizations will jointly optimize for productivity and skills development among their employees. In fact, in some of my research, great productivity was only achieved by insisting on simultaneous skill development. Measuring and driving for both is the path to sustainable performance.

AI is an integral part of the future of software development—and that includes skill development. For example, we can use AI to parse developer–AI interactions, link this to version control interactions, and then to skill and key work outcomes.

That wasn't cost-effective before, but API costs for labeling are falling rapidly. With AI we can get the insight we need to refactor the work and offer guidance to developers on how they can keep their learning edge—and help others do the same.

Default AI usage patterns are delivering breakthrough productivity and blocking skill development for most devs. To keep our innovative edge—both individually and collectively—we need to use AI itself to measure skill development and productivity simultaneously.

Metrics frameworks

Sarah D'Angelo, Ph.D.
User Experience Researcher, Google

Ambar Murillo
User Experience Researcher, Google AI & Infrastructure

Sarah Inman, Ph.D.
User Experience Researcher, Google

Kevin M. Storer, Ph.D.
User Experience Researcher, Google Cloud

Choosing measurement frameworks to fit your organizational goals

Measuring software development can help drive impactful change. However, it's a complex task, and getting started can be daunting as it involves understanding what you should measure, and determining what you can measure.

The most important part is that you want to drive change in your organization through measurement. To do so, we recommend that you use frameworks to guide your measurement strategy.

A framework breaks down a broad topic (for example, developer experience) into distinct concepts that can be measured (such as speed or satisfaction).

When industry and academia talk about measuring aspects of software development, they often reference frameworks such as SPACE,[1] DevEx,[2] HEART,[3] and DORA's software delivery metrics.[4] Choosing a framework to measure software development can be a difficult and confusing step, but it doesn't have to be.

The first step is to define what goals and decisions measurement will inform, because frameworks differ in their overarching goals. For example, common frameworks in software development focus on measuring developer experience, product excellence, and organizational effectiveness. Each of these overarching goals take a slightly different lens to understanding software development (see Figure 52).

Types of frameworks typically applied to measuring software development

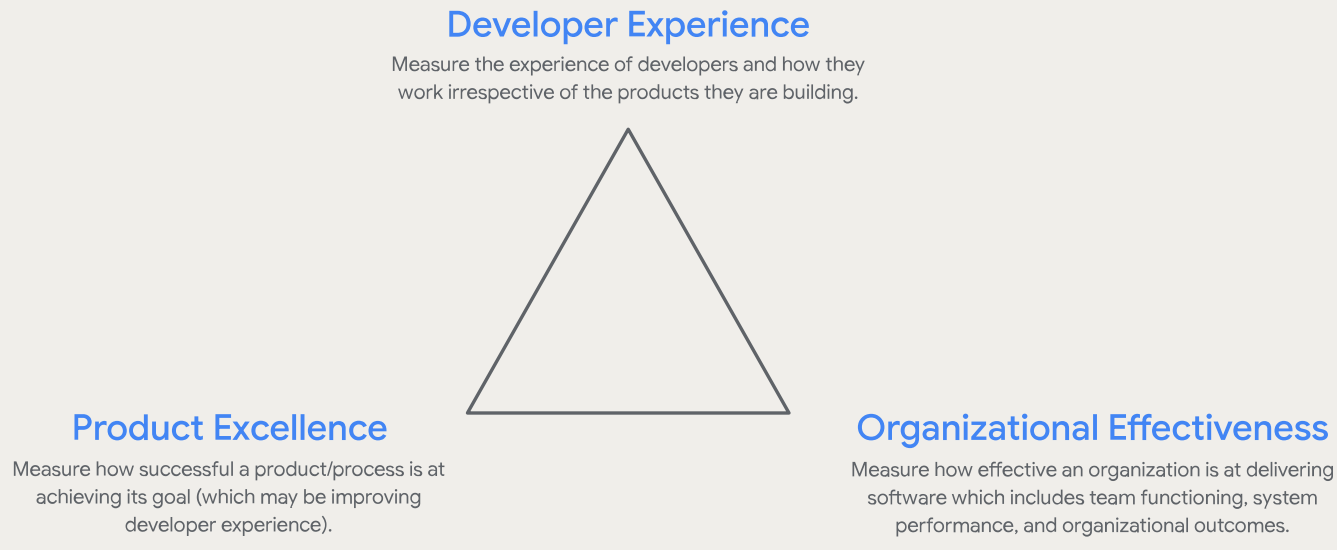


Figure 52: Types of frameworks typically applied to measuring software development

To determine which framework is the best fit for your organization's goals, it can be helpful to think of frameworks as the "why" behind measurement. They help you define why you're trying to measure and guide the actions you take based on your findings.

Frameworks provide a lens through which to view your data, ensuring your efforts are aligned with your organizational goals. To decide on a framework, you can consider: Why now? Did something change that is motivating a desire to measure? How will you act on your insights? Are there decisions or improvements you can enable with measurement?

Now, the "what" you'll measure is the actual metrics, the key concepts that contribute to the overarching framework, such as a velocity or adoption metric. Generally, there are two different approaches to data collection you can take. This is the "how" of your data collection which will help you arrive at your metrics.

Self-reported data involves collecting information directly from developers about their experiences. This can be achieved through approaches such as:

- Surveys use questions to gather opinions, satisfaction levels, and perceptions on various aspects of work.
- Interviews and focus groups use one-on-one and group discussions to go deeper into specific experiences and topics.
- Diary studies collect in situ data about activities, thoughts, and experiences.

The strength of self-reported data is in its ability to capture subjective experiences and concepts that are difficult to quantify through automated means, such as satisfaction, well-being, or perceived effectiveness. A key advantage of self-reported data is that it doesn't typically require extensive instrumentation or deep observability into developers' toolchains.

However, self-reported data does present challenges in terms of standardization, comparability across teams, and scalability for large organizations. Its inherent subjectivity means that interpretations can vary, and it may be more susceptible to biases (including recall bias and social desirability bias).

Logs-based measures are collected automatically from the tools and systems developers use. These can include metric types such as:

- Quantity, to count specific artifacts. For example, the number of commits or number of users.
- Time-based, to record how much time is spent on a given activity. For example, the time spent coding or reviewing.
- Frequency, measuring the rate over a specific window of time. For example, monthly deployments, or weekly PRs per developer.

The primary benefit of logs-based measures is their ability to provide continuously measured and standardized data at scale. They offer a detailed view of activities and outputs.

However, they require sufficient observability into the developer toolchain, meaning the necessary integrations and data collection mechanisms must be in place, which can create a higher barrier to entry.

It is also a common misconception that logs-based metrics are objective. Instrumentation approaches vary, errors can create inaccuracies, and interpretation is subject to bias.

A framework will provide you with the concepts you want to measure, but ultimately what you implement depends on your resourcing, and what data you have access to. Do you have the visibility into your toolchain for logs-based approaches, or a research team to enable self-reported data collection? It's important to recognize that not all organizations have the same resourcing and ability to implement metrics in the ways that might be recommended by a particular framework.

Even with organizational limitations, frameworks are a guide, a lens to help you better understand complex behavior—but they can not fully capture it. They're intended to get you closer to the truth, but you shouldn't expect to measure everything.

When considering how frameworks and metrics relate, it's helpful to think of metrics as ingredients, and frameworks as the recipe that is made with the ingredients. Some core ingredients can be rearranged in different ways to make different recipes (the frameworks), while others are unique to a specific recipe. The meals will all taste different but some of the underlying ingredients are shared, and in many cases you don't need to have all of the ingredients to make a meal that tastes good.

While frameworks differ because they are intended to drive different outcomes, some of their underlying metrics overlap. The diagram below illustrates some of the metrics that comprise frameworks and how they often overlap. For example, productivity metrics (such as code commits or pull requests) may be measured by all three frameworks.

Examples of metrics that apply to different frameworks

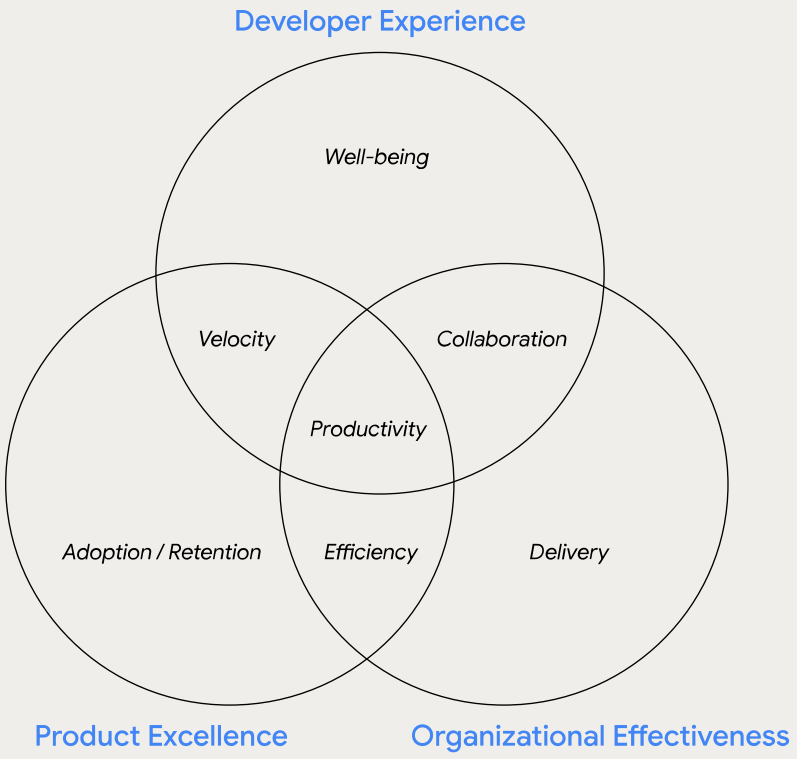


Figure 53: Examples of metrics that apply to different frameworks

An organization might use these metrics to gauge the impact of a new team structure (organizational effectiveness), evaluate the effectiveness of a developer tool (product excellence), and understand developer workload (developer experience).

In contrast, some metrics are more specialized. Developer well-being, often a key component of developer experience frameworks, is not typically a primary metric within organizational effectiveness or product excellence frameworks.

Choosing to use a single framework can help provide focus to the actions you take, and it is a good way to start. However, you're not limited to one framework.

As goals and abilities to measure change, using multiple frameworks can help create complementary analytical results,[5] resulting in a stronger whole than its parts. What matters is that you're measuring as a way to hold yourself and your organization accountable to your goals, and that you are positioned to act on what you measure.

Applying measurement frameworks in the age of AI

You might be wondering, does the introduction of AI into development workflows change everything? Do the same frameworks apply or do we need new frameworks? When there is a technological disruption, it may seem necessary to completely overhaul your metrics collection strategy. We recommend careful consideration of what actually needs to be changed, especially when considering the impact of AI.

Adapting your goals to better understand how AI is impacting developer experience may require updating only a few metrics, allowing you to retain consistent measures overall. Instead of throwing out the entire framework, you can use existing measures as a baseline to help identify how a paradigm shift is changing the developer experience. For example, you may need to add metrics on the acceptance rates of AI suggestions, model quality, or trust, while keeping existing measures of developer experience, such as perceived productivity and time spent reviewing code.

As we see more substantial advances in AI, who is doing development tasks—and what those are—will change. So, measurement may have to adapt to include different user profiles and capture changing workflows, but the core goals behind why you are measuring developer experience likely haven't changed. The point here is that if your overarching goal is the same, you don't need to change your framework; you can expand your measurements to adapt to changes in technology.

Even if your goals do change, it does not necessarily result in starting a measurement program from scratch. Since metrics can contribute to different frameworks, you often can react quickly and rearrange the metrics in the service of new or additional frameworks. For example, understanding the impact of AI-powered developer tools on the code that is being produced is likely a new goal that organizations have not faced before. This is particularly challenging because we are trying to measure something as it is changing.

A common question organizations are facing is the impact of AI on code quality.[6] As we see AI used to increase developer velocity, there is a particular concern that we are compromising quality for speed. These increases in developer velocity in the short term can seem positive; however, they may have negative impacts on velocity in the long term if quality is low.

To address these concerns, your goal might be to ensure the code quality at your organization remains high while you drive adoption of AI-powered tooling. This goal involves aspects from each of the types of frameworks discussed and likely includes metrics you are already capturing (such as code quality, tool adoption, or perceived velocity).

So, you can continue to use your existing metrics while introducing new ones. For example, combining DORA's software delivery metrics with a product excellence framework such as HEART. can be an effective way to understand how developers are experiencing new AI-powered tools and the impact on software delivery.[7]

Measuring software development is a complex and ongoing process. While many frameworks and measurement approaches are available, you must be positioned to act on what you measure. A critical aspect of ensuring effective action can be taken is aligning with your organization's goals and getting leadership sponsorship for your measurement efforts.

Being intentional about the framework and measurements you choose can help set you up for long-term success. In the spirit of adopting a framework to meet a specific goal, you can consider how you might act on this information following the PDCA framework:

- **Plan:** figure out your goals, choose a framework, and get leadership support
- **Do:** get some baseline measures, do something differently
- **Check:** measure again to see how you're progressing towards your goals
- **Adjust:** use your findings to change the approach moving forward

We are not here to recommend one framework over others. Determining the appropriate framework based on your goals can help guide what you measure and how you take action. Choose the framework that resonates with your organization. If it speaks to you and spurs your organization into action, it's the right framework for right now.

While the frameworks provide a guiding structure, many of the underlying measures are the same. This means that measures you implement today can often be adapted and utilized as your needs and goals evolve, or change.

1. "The SPACE of Developer Productivity: There's more to it than you think." https://queue.acm.org/detail.cfm?id=3454124
2. "DevEx: What Actually Drives Productivity: The developer-centric approach to measuring and improving productivity." https://queue.acm.org/detail.cfm?id=3595878
3. "Measuring the User Experience on a Large Scale: User-Centered Metrics for Web Applications." https://research.google/pubs/measuring-the-user-experience-on-a-large-scale-user-centered-metrics-for-web-applications
4. "DORA's software delivery metrics: the four keys." https://dora.dev/guides/dora-metrics-four-keys
5. "Unlocking product success by combining DORA and H.E.A.R.T." https://cloud.google.com/transform/unlocking-product-success-by-combining-dora-and-heart
6. "Measuring AI code assistants and agents." https://getdx.com/research/measuring-ai-code-assistants-and-agents
7. "Unlocking product success by combining DORA and H.E.A.R.T." https://cloud.google.com/transform/unlocking-product-success-by-combining-dora-and-heart

Final thoughts: From insight to action

Nathen Harvey
DORA Lead, Google Cloud

For over a decade, DORA has been a trusted partner in the software development community, providing research and insights to help teams improve. As the industry rapidly evolves with the adoption of new technologies like AI and platform engineering, our commitment remains the same: to investigate and share the practices that foster high-performing teams.

DORA AI Capabilities Model

This year, we introduced the inaugural DORA AI Capabilities Model, a significant evolution of our research. As organizations navigate the complexities of AI adoption, this model provides a data-backed framework to guide their journey. It highlights seven critical capabilities that, when cultivated, amplify the positive impacts of AI on important organizational outcomes.

These capabilities are:

- A clear and communicated AI stance
- Healthy data ecosystems
- AI-accessible internal data
- Strong version control practices
- Working in small batches
- A user-centric focus
- Quality internal platforms

This model is our first iteration, and we consider it a starting point for an ongoing conversation with the DORA community and organizations embracing AI-assisted software development. We are eager to learn from your experiences as you apply these insights and look forward to validating and refining the model in our future research.

Focus on the user

This year's research reveals a critical insight: We're still in the nascent stages of AI-assisted software development, a period of rapid technological and practical evolution where early standardization would be premature. Our findings show that simply deploying AI tools is not a panacea for transformation. In fact, AI's impact on team performance is strikingly dependent on a crucial factor: a user-centric focus.

We found with a high degree of certainty that when teams adopt a user-centric focus, the positive influence of AI on their performance is amplified. Conversely, in the absence of a user-centric focus, AI adoption can have a negative impact on team performance.

This finding underscores a vital message for all organizations: Investing in and cultivating a deep understanding of your end users is not just beneficial, it's a prerequisite for success with AI. Without a user at the center of your strategy, AI adoption is unlikely to help, and may even hinder, your team's performance.

Putting research into practice: Your turn to experiment

The findings in this year's report are complex, and at times may even appear contradictory. This reflects the reality of a field in flux. We encourage you to treat our findings not as rigid prescriptions, but as hypotheses for your own experiments.

Here are some ways you can put this research into practice:

- **Run experiments in your organization:** Use DORA's findings to formulate hypotheses and test them on your teams. This will allow you to learn more about your specific operational context and identify the most impactful areas for improvement.
- **Conduct internal surveys:** Take inspiration from the questions used in this year's survey[1] to design your own. Tailor them with more nuanced questions that are directly relevant to your teams and projects.
- **Embrace the holistic experience:** Remember that improving a single feature won't fix a flawed platform. Treat your internal platforms as products, focusing on the entire developer journey from feedback loops to automation.
- **Share what you learn:** As you conduct your own experiments and gather insights, disseminate that knowledge throughout your organization. This can be through formal reports, informal communities of practice, or casual conversations. The goal is to foster a culture of continuous learning.

The greatest risk in this evolving landscape isn't falling behind, but rather investing heavily in chaotic activity that fails to deliver meaningful results. Choose the frameworks and approaches that resonate with your organization and spur you to action.

Join the conversation

Thank you for engaging with our research. We invite you to continue the journey with us. Share your experiences, learn from your peers, and find inspiration by joining the DORA community at https://dora.community.



1. "DORA Research: 2025 Survey questions." https://dora.dev/research/2024/questions

Acknowledgements

The DORA research is a testament to the power of a passionate and collaborative global community. We are immensely grateful to the thousands of researchers, experts, practitioners, leaders, and transformation agents who so generously share their knowledge and experiences. This report is a direct result of that collective effort. Our thanks also go to you, the reader. We hope these findings will empower you and your teams to drive meaningful change and continuous improvement within your organizations.

Research partners

The 2025 DORA Report is presented by Google Cloud and IT Revolution in collaboration with the following research partners.

Presented by

Google Cloud

IT REVOLUTION

Research partners

GitHub GitLab

skillbench Workhelix

Sponsors

The 2025 DORA Report was supported by the following sponsors. Learn more about these sponsors on the DORA website.[1]

Platinum sponsors

swarmia

thoughtworks

Gold sponsors

Buildkite CodeRabbit

DATADOG Deloitte.

harness JELLYFISH

Octopus Deploy OPSERA

Silver sponsors

Allstacks ATLASSIAN

Checkmarx chronosphere

COPIA hivel

honeycomb.io

DORA Report team

Matt Beane, Ph.D.

James Brookbank

Kim Castillo

Sarah D'Angelo, Ph.D.

Derek DeBellis

Rob Edwards

Edward Fraser

Benjamin Good

Nathen Harvey

Sarah Inman, Ph.D.

Eirini Kalliamvakou, Ph.D.

Gene Kim

Eric Maxwell

Ambar Murillo

Allison Park

Daniel Rock, Ph.D.

Kevin M. Storer, Ph.D.

Daniella Villalba, Ph.D.

Steve Yegge

Jessie Young

Editor

Seth Rosenblatt

Accuracy Matters[2]

DORA Community guides

Dhruv Agarwal

Lisa Crispin

Steve Fenton

Denali Lumma

Betsalel (Saul) Williamson

Advisors and experts in the field

Ameer Abbas

Colette Alexander

Christy Barrett

Sander Bogdan

Kevin Borders

Adam Brown

Michele Chubirka

Andrew Davis

Blanca Delgado

Thomas De Meo

Nicole Forsgren, Ph.D.

Vivian Hu

Jez Humble

Michelle Irvine

Ben Jose

Deepti Kamat

Amanda Lewis

Mike Long

Steve McGhee

Rebecca Murphey

Jacek Ostrowski

Justin Reock

Willy Rouvre

Ryan J. Salva

Harini Sampath

Dadisi Sanyika

Robin Savinar

Sean Sedlock

Jerome Simms

Dustin Smith

Dave Stanke

Laura Tacho

Adrienne Braganza Tacke

Finn Toner

Cedric Yao

Members of the DORA community

Research and design partners

Apparent:[3] DORA branding

Human After All:[4] DORA report design

Prolific:[5] Research infrastructure support

User Research International:[6] Research infrastructure support

1. "DORA 2025 Sponsors." https://dora.dev/research/2025/sponsors
2. "Accuracy Matters." https://accuracymatters.co.uk
3. "We Are Apparent." https://apparent.com.au
4. "Human After All: Clarity through creativity." https://www.humanafterall.studio
5. "Prolific | Easily collect high-quality data from real people." https://www.prolific.com
6. "User Research for Product Development | User Research International." https://www.uriux.com

Authors



Derek DeBellis

Derek DeBellis has been leading DORA's research program since 2022, now happily co-leading alongside Dr. Kevin M. Storer.

He is driven by a central question: What helps people work happily and effectively together? This question motivates his work at DORA and his independent research into power dynamics, learning architectures, and belief propagation.

When he's not analyzing data, Derek can be found trail running (twisting his ankles) in the Colorado Rockies, reading Kafka and Borges, or recording music.



Kevin M. Storer

Dr. Kevin M. Storer leads ethnographic research for Google Cloud's DORA team.

His work focuses on understanding how organizations can best adapt their software development practices in response to recent advancements in gen AI, especially as it is used in coding.



Nathen Harvey

Nathen Harvey leads Google Cloud's DORA team, using its research to help organizations improve software delivery speed, stability, and efficiency.

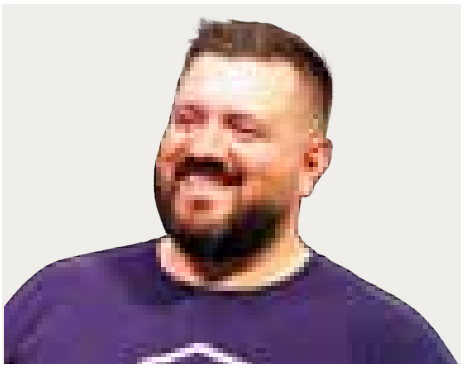
He focuses on enhancing developer experience and is dedicated to fostering technical communities like the DORA Community, which provides opportunities to learn and collaborate.



Matt Beane, Ph.D.

Matt Beane is an Associate Professor in Technology Management at the University of California, Santa Barbara, Digital Fellow at Stanford and MIT, and CEO/Cofounder of SkillBench, a firm that helps companies boost productivity and skills through gen AI use.

He has studied the deployment of intelligent technologies since 2011. He is author of *The Skill Code: How to Save Human Ability in an Age of Intelligent Machines* (HarperCollins, 2024).



Rob Edwards

Rob Edwards is a Lead for Software Delivery and Developer Experience at Google Cloud.

He partners with customers to help their engineering teams build and ship software more smoothly and reliably.

He works from the belief that the best technical solutions come from people working together toward a shared goal, making technology a tool for business success.



Edward Fraser

Edward Fraser is a graduate student at UC Berkeley's School of Information, exploring how human–computer interaction and emerging technologies can create more intuitive, empowering experiences.

Drawing on professional experience in design and product management, his recent work spans an eye-tracking study of developers using AI coding assistants, UX research for enterprise AI tools, and the design of an educational platform for incarcerated learners.

Authors



Benjamin Good

Benjamin Good is a Cloud Solutions Architect at Google.

He is passionate about improving software delivery practices through cloud technologies and automation.

As a Solutions Architect he gets to help Google Cloud customers solve their problems by providing architectural guidance, publication of technical guides, and open source contributions. Prior to joining Google, Ben ran cloud operations for several companies in the Denver/Boulder area, implementing building platforms along the way.



Eirini Kalliamvakou

Dr. Eirini Kalliamvakou is a Staff Researcher turned Research Advisor to GitHub's CEO. Eirini's focus is on understanding developers' needs, motivations, and behavior.

Her work informs evaluations, innovation strategy, and storytelling at GitHub, and her award-winning research on AI tools and DevEx is shaping industry practice and community influence.



Gene Kim

Gene Kim is an award-winning researcher and author. He has been studying high-performing technology organizations since 1999.

His books have sold over 1 million copies—he is a WSJ bestselling author of *The Unicorn Project*, and co-author of *The Phoenix Project*, *The DevOps Handbook*, and two Shingo Publication Award-winning books, *Accelerate* and *Wiring the Winning Organization*.

In 2025, he won the Philip Crosby Medal from the American Society for Quality (ASQ). His latest book, *Vibe Coding*, is co-authored with Steve Yegge (coming October 2025).

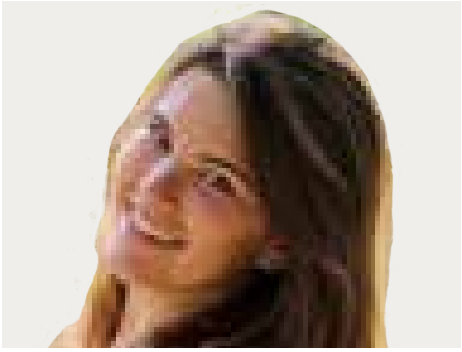




Eric Maxwell

With a career that began in the trenches of software engineering, Eric Maxwell has always been driven by a passion for automation and a strong empathy for fellow practitioners. This foundational experience is the cornerstone of his work today as the leader of Google's 10x Technology consulting practice.

Eric advises top global companies on how to deliver value faster by transforming their organizations through strategic improvements in technology, process, and culture. He contributes his expertise to the renowned DORA team. Before his time at Google, Eric was at Chef Software, where he honed his skills in infrastructure automation and whipping up awesome.



Sarah D'Angelo, Ph.D.

Sarah D'Angelo is a Senior Staff UX Researcher on Google's Core Labs team.



Sarah Inman

Sarah Inman is a Senior UX Researcher on Google's Chrome team.



Ambar Murillo

Ambar Murillo is a Staff UX Researcher on Google's AI & Infrastructure team.



Daniella Villalba, Ph.D.

Daniella Villalba is a Senior UX Researcher on Google Cloud. Her research focuses on understanding developers.

Demographics and firmographics

The DORA research program has been researching the capabilities, practices, and measures of high-performing, technology-driven organizations for over a decade. In that time, we've heard from roughly 45,000 professionals working in organizations of every size and across many different industries.

This year, nearly 5,000 working professionals from a variety of industries around the world shared their experiences to help grow our understanding of how AI is changing the field of software development and the conditions in which organizations can continue to thrive in this new paradigm. In this chapter, we share a bit more about our respondents, their identities, their geographies, and their role in software development.

We thank each of you for sharing your insights with us!

Survey respondent demographics

This year, a total of 4,867 respondents answered our survey,[1] across a range of demographics, locations, and industries.

Individual

Age

We asked respondents their age, as an open-text response. Our sample had a median age of 41.

Distribution of respondents age

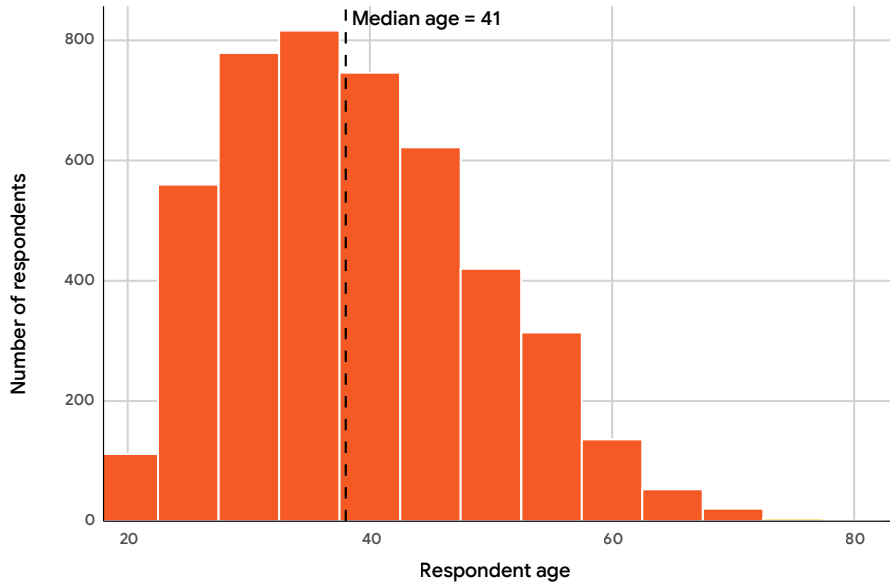


Figure 54: Distribution of respondents age

Gender

We asked survey respondents to report their gender. 62.8% of respondents identified as men, 19.2% as women, 1.1% identified as non-binary, 0.5% chose to self-describe, and 2.1% declined to answer.

Distribution of respondents gender

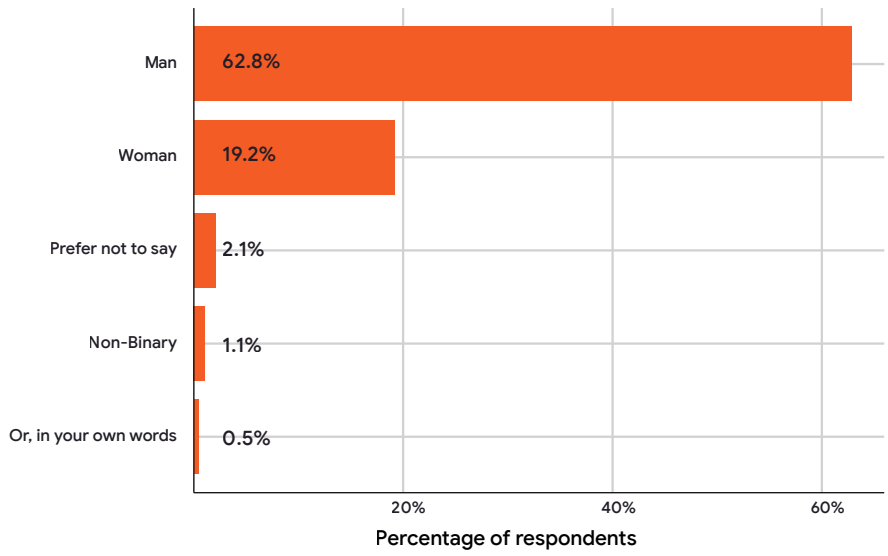


Figure 55: Distribution of respondents gender

Disability

We identified disability along six dimensions that follow guidance from the Washington Group Short Set.[2] This is the sixth year we have asked about disability.

Distribution of self-reported disabilities

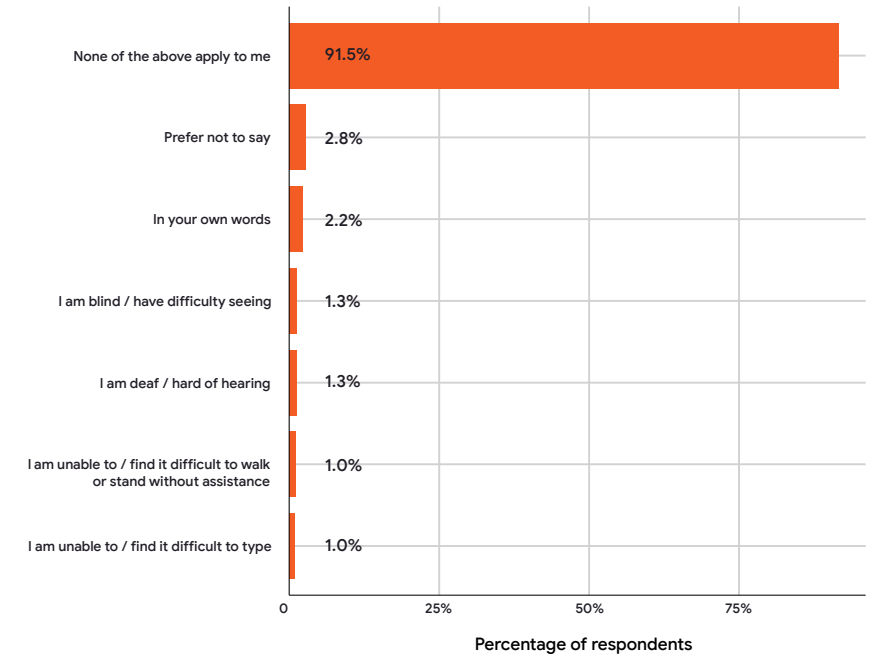


Figure 56: Distribution of self-reported disabilities

Roles

We include a large number of roles in our survey to capture the numerous ways that someone might be involved in software development. The most highly represented roles in our data are full-stack developers, engineering managers, and back-end developers.

Distribution of respondents role

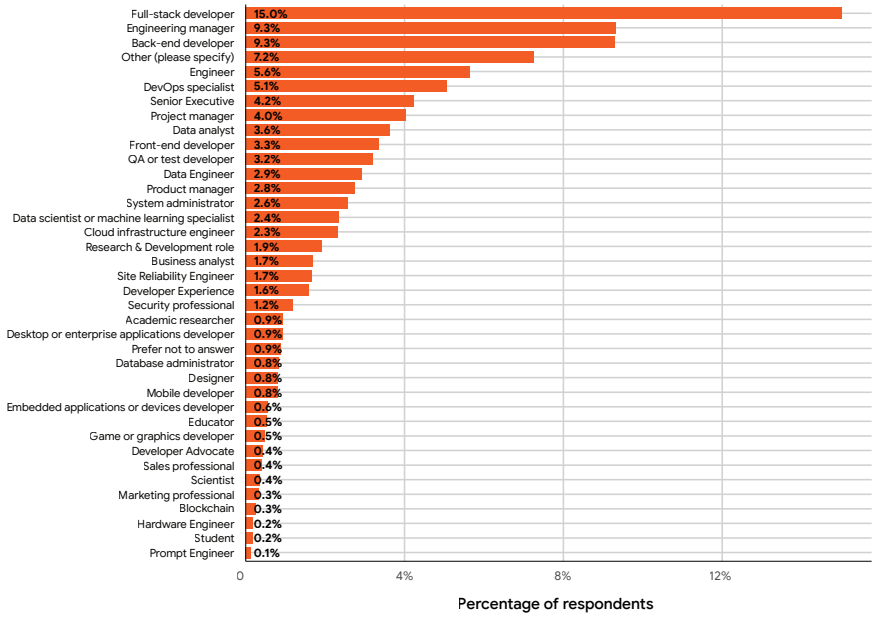


Figure 57: Distribution of respondents role

Role experience

We asked survey respondents to report their years of experience in their role or similar positions. Respondents had a median of six years of experience working in their role.

Distribution of respondents role experience

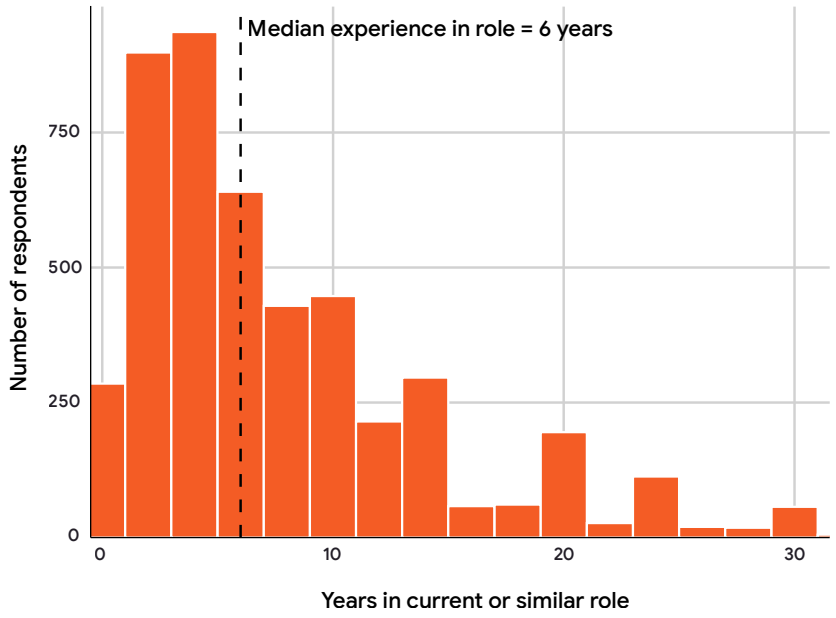


Figure 58: Distribution of respondents role experience

Team experience

We asked survey respondents to report their years of experience in their current team. Respondents had a median of three years of experience working on their team.

Distribution of respondents team tenure

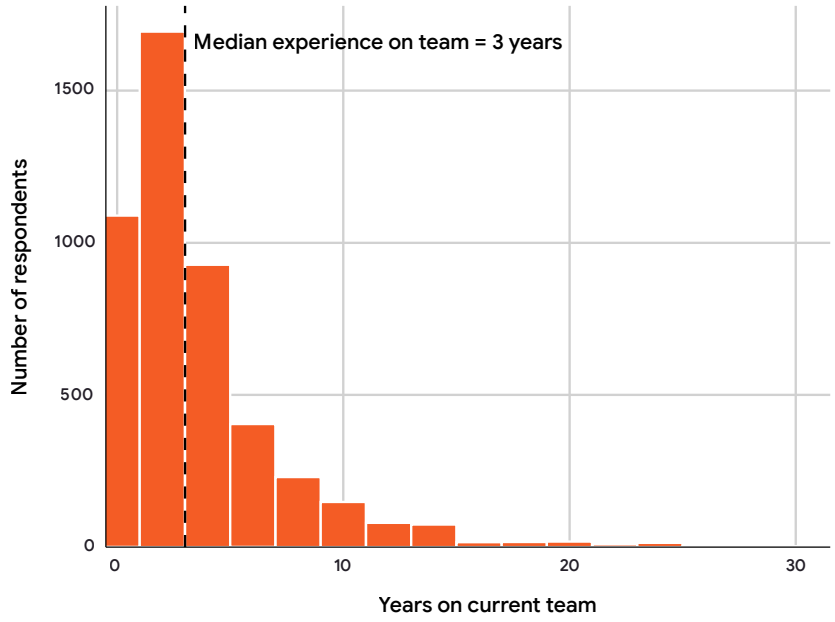


Figure 59: Distribution of respondents team tenure

Where they work

We asked respondents approximately what percentage of their last five days were spent working in an office. Despite broader initiatives to return workers to physical office spaces, 800 of our respondents worked their most recent five days fully remotely. The median time spent in an office over the last five days was 50%, suggesting a hybrid model is most common.

Distribution of time spent in office

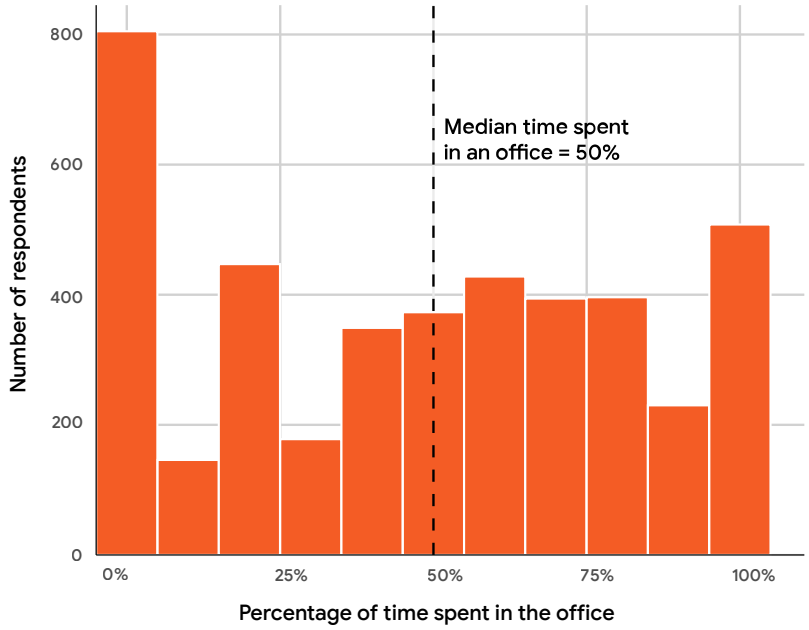


Figure 60: Distribution of time spent in office

Coding language

We asked respondents which programming languages they used most frequently at work, supporting up to three top choices. Almost half of respondents are using Python at work, while about one-third are using each of JavaScript and SQL.

Programming language usage

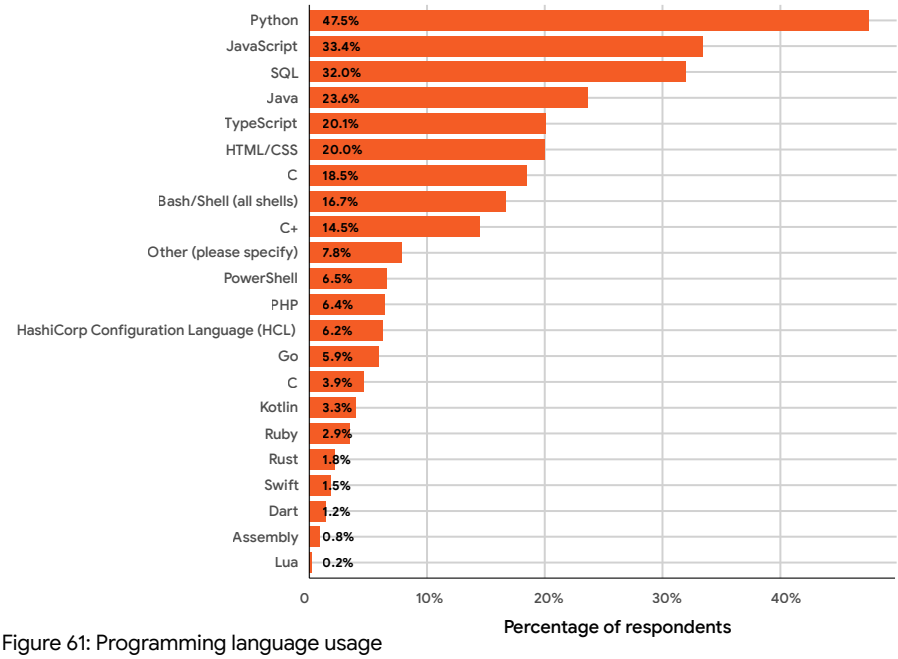


Figure 61: Programming language usage

Organization

Industry

We asked survey respondents to identify the industry sector in which their organization primarily operates, across 12 categories. Excluding "Other" responses, the most common sectors in which respondents worked were Technology, Financial Services, and Retail/ Consumer/E-commerce. This matched our top three 2024 industry demographics.

Distribution of respondent industries

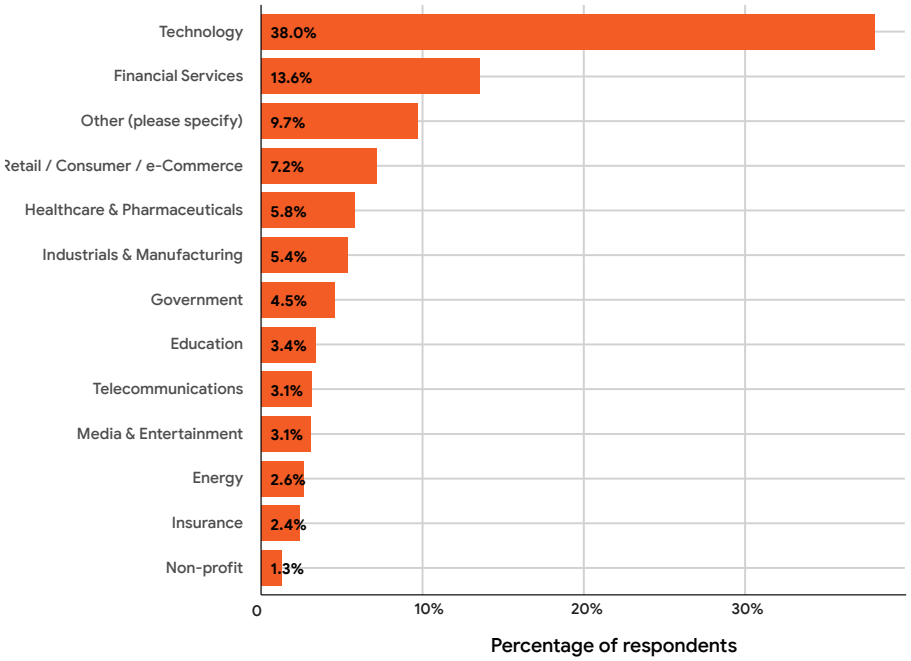


Figure 62: Distribution of respondents industries

Size

We asked survey respondents to identify the number of employees at their organization, using nine buckets. The organizations in which respondents worked most commonly had 10,001 or more employees (22.4%), 51 to 200 employees (16.2%), and 1,001 to 5,000 employees (14.5%).

Distribution of organization sizes

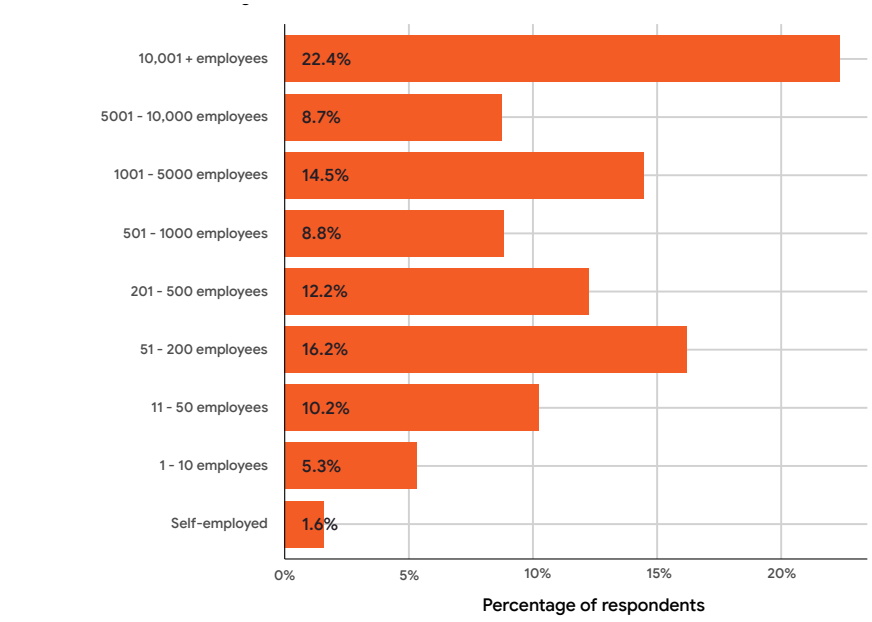


Figure 63: Distribution of organization sizes

Service

AI-infused services and applications

We asked respondents to indicate whether they agreed that their application or service was actively adding AI-powered experiences across the last three months. A roughly equal number of respondents agreed and disagreed, with more than a quarter strongly disagreeing that their application was adding AI-powered experiences in this timeline.

Active integration of AI into end-user service

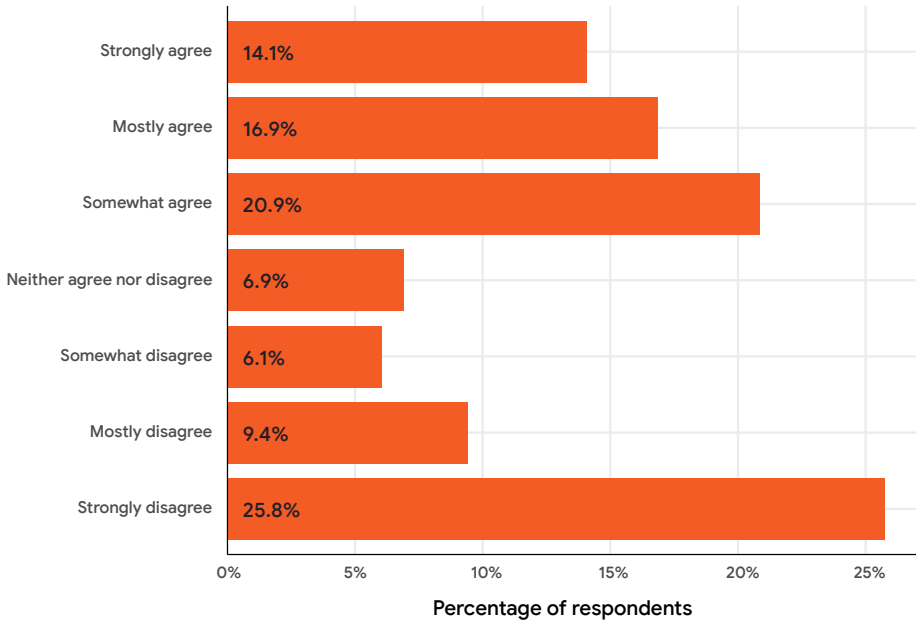


Figure 64: Active integration of AI into end-user service

Application criticality

We asked respondents to indicate the criticality of their application to their employer by indicating what level of impact the unavailability of that application would have for the organization's ability to achieve its goals and serve its customers. More than half of our respondents felt the unavailability of their application would have "a great deal" of impact on the company.

Impact of primary service unavailablity

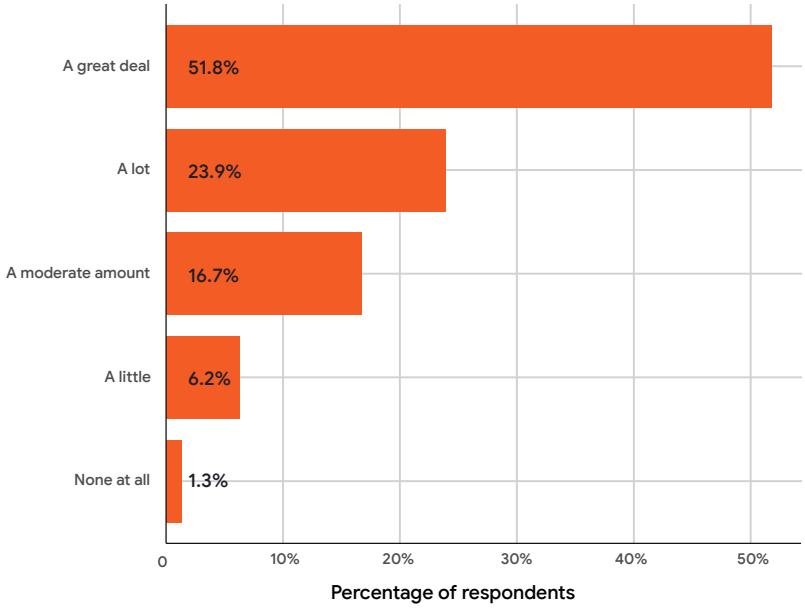


Figure 65: Impact of primary service unavailablity

Service age

We asked participants to indicate approximately how many years the primary application or service they work on has existed. Our respondents' applications had a median age of eight years.

Distribution of primary service or application age

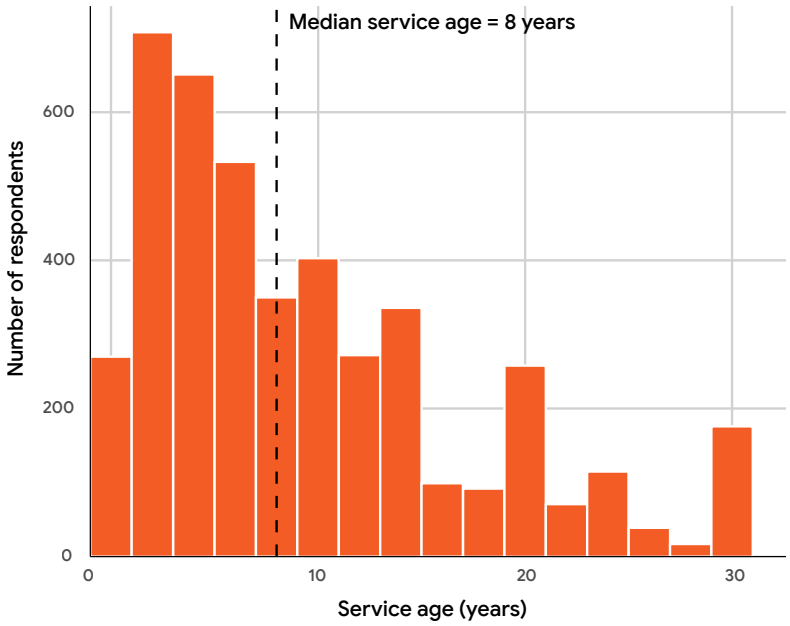


Figure 66: Distribution of primary service or application age

Service users

We asked participants to indicate characteristics of their applications' primary end users. A majority of respondents were developing business applications, with a roughly even number developing for internal and external audiences.

Primary service end-user characteristics

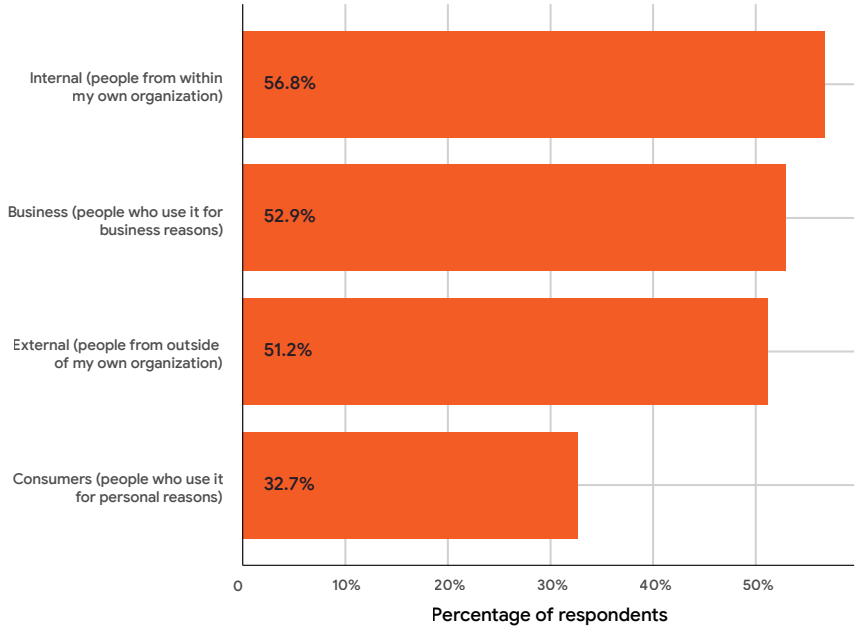
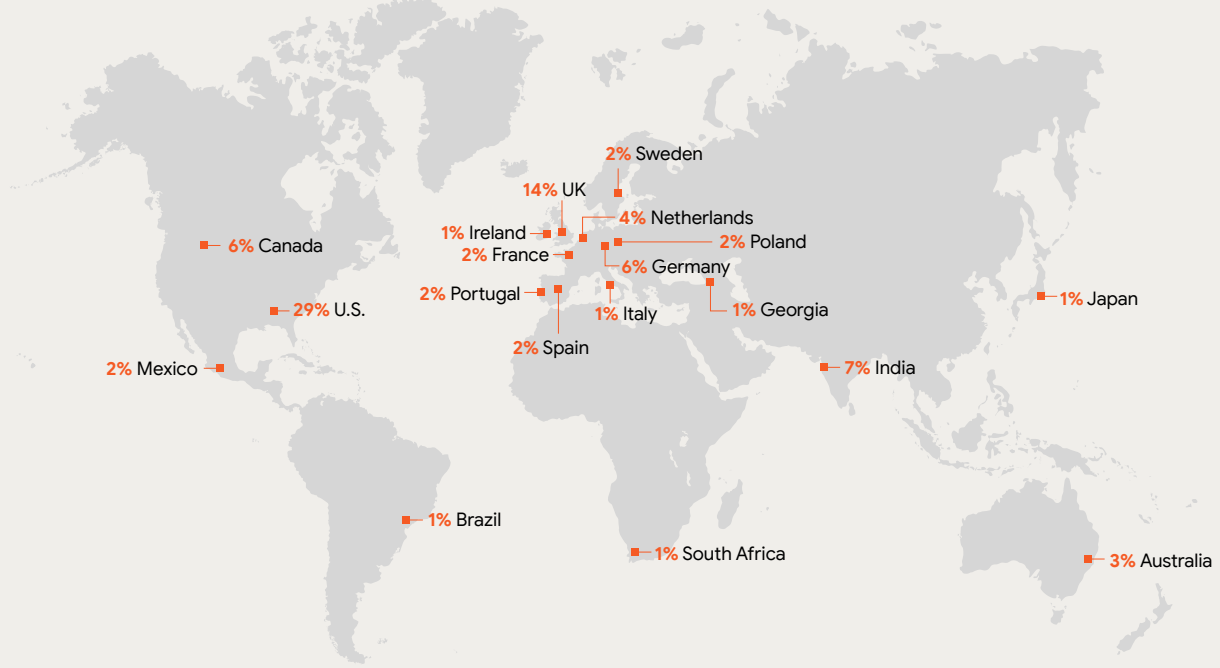


Figure 67: Primary service end-user characteristics

Country

This year, we had respondents from more than 100 countries. Our largest number of respondents were located in the U.S., followed by the UK, and India.



Country

USA	Japan	Malaysia	Peru	Uruguay	Madagascar
UK	New Zealand	Romania	Serbia	Viet Nam	Morocco
India	Switzerland	Estonia	Slovakia	Andorra	Nicaragua
Germany	Hungary	Slovenia	Thailand	Armenia	Panama
Canada	Belgium	Indonesia	Ukraine	Bahamas	Paraguay
Netherlands	Denmark	United Arab Emirates	Afghanistan	Bahrain	Republic of Moldova
Australia	Norway	Bulgaria	Ecuador	Belarus	Somalia
France	Ireland	Croatia	El Salvador	Bolivia	Sri Lanka
Portugal	Chile	Costa Rica	Jordan	Burkina Faso	The former Yugoslav Republic of Macedonia
Poland	Philippines	Nigeria	Kenya	Democratic Republic of the Congo	Tunisia
Mexico	Greece	Russian Federation	Lithuania	Ethiopia	Uganda
Sweden	Israel	Turkey	Malta	Grenada	Yemen
Spain	Singapore	Bangladesh	South Korea	Hong Kong (S.A.R.)	Zambia
NA	Finland	China	Algeria	Iceland	
Georgia	Austria	Costa Rica	Cyprus	Iran	
Italy	Czech Republic	Egypt	Côte d'Ivoire	Kazakhstan	
South Africa	Argentina	Latvia	Dominican Republic	Lebanon	
Brazil	Colombia	Pakistan	Guatemala	Luxembourg	

Color intensity indicates the number of survey respondents from each country, with darker shades representing a higher volume of participants.

Interview participant demographics

The sole inclusion criterion for our interviews is that participants are involved in some way in professional software development. Our screener collects no information about participants' demographics outside of those required to confirm their job, location, and language eligibility. In total, we interviewed 78 participants who met these criteria.

When asked about their responsibilities, 70 interview participants indicated that they personally write or modify source code, 37 indicated that they manage software delivery pipelines and/or development infrastructure, 15 indicated that they make purchasing decisions about development products and services, 12 indicated that they define and update organizational policies about technology use, and two indicated that their work related to software development only in some other way.

Although these responsibilities suggest their roles were typically multifaceted, we asked participants how they would best describe their work from the following options: "I am a software developer", "I administrate software development infrastructures", "I manage people who develop software", "I set policies about software development for my organization", "I make purchasing decisions about products and services used in software development", "My work is not at all related to software development", "My work is related to software development in a way that is not listed here".

67 interviewees described themselves as primarily software developers, seven described themselves as primarily managers of software developers, one indicated being primarily an administrator of software development infrastructures, and three indicated their work primarily relates to software development in some other way.

76 interviewees were located in the U.S., one was located in Mexico, and one was located in Trinidad and Tobago. That the vast majority of participants were located in the U.S. was not surprising, given the interviewer's language fluency and scheduling limitations.

1. Not every respondent was part of each analysis because of survey conditionality or missingness.
2. "WG Short Set on Functioning (WG-SS)." https://www.washingtongroup-disability.com/question-sets/wg-short-set-on-functioning-wg-ss/

Methodology

A methodology is supposed to be like a recipe that will help you replicate our work and determine if the way our data was generated and analyzed is likely to return valuable information. Although we don't have the space to go into the exact details, hopefully this is a great starting point for those considerations.

Derek DeBellis

Quantitative User Experience Researcher, Google Cloud

Kevin M. Storer, Ph.D.

User Experience Researcher, Google Cloud

Survey development

Question selection

We think about the following aspects when considering whether to include a question in a survey:

Is this question...

- **Established so we can connect our work to previous efforts?**
- **Capturing an outcome the industry wants to accomplish (for example, high team performance)?**
- **Capturing a capability the industry is considering investing resources into (for example, AI)?**
- **Capturing a capability we believe will help people accomplish their goals (for example, quality documentation)?**
- **Something that helps us evaluate the representativeness of our sample (for example, role or gender)?**
- **Something that helps us block biasing pathways (for example, coding language or role)?**
- **Something that is possible to answer with at least a decent degree of accuracy for the vast majority of respondents?**

We address the literature, engage with the DORA community, conduct cognitive interviews, run parallel qualitative research, work with subject matter experts, and hold team workshops to inform our decision as to whether to include a question in our survey.

Survey experience

We take great care to improve the usability of the survey. We conduct cognitive interviews and usability tests to make sure that the survey hits certain specification points:

- **Time needed to complete survey should, on average, be low**
- **Comprehension of the questionnaire should be high**
- **Effortfulness should be reasonably low, which is a huge challenge given the technical nature of the concepts**

Data collection

Localizations

People around the world have responded to our survey every year. This year we worked to make the survey more accessible to a larger audience by localizing the survey into English, Español, Français, Português, 日本語, and 简体中文.

Collect survey responses

We use multiple channels to recruit. These channels fall into two categories: organic and panel.

The organic approach is to use all the social means at our disposal to let people know that there is a survey that we want them to take. We create blog posts. We use email campaigns. We post on social media, and we ask people in the community to do the same (that is, snowball sampling).

We use the panel approach to supplement the organic channel. Here we try to recruit people who are traditionally underrepresented in the broader technical community and try to get adequate responses from certain industries and organization types.

In short, this is where we get some control over our recruitment—control we don't have with the organic approach. The panel approach also allows us to simply make sure that we get enough respondents, because we never know if the organic approach is going to yield the responses necessary to do the types of analyses we do. This year we had sufficient organic responses to run our analysis and the panel helped round out our group of participants.

Survey flow

This year we had a lot of questions we wanted to ask, but not enough time to ask them. Our options were...

- **Make an extremely long survey**
- **Choose a subset of areas to focus on**
- **Randomly assign people to different topics**

We didn't want to give up on any of our interests, so we chose to randomly assign participants to one of four separate flows. There was a lot of overlap among the four different flows, but each flow dove deeply in a different space.

Here are the four different pathways:

- **AI**
- **Platform engineering**
- **Sociocognitive aspects**
- **AI capabilities**

Interviews

Last year we introduced the use of in-depth, semi-structured interviews to supplement our annual survey with qualitative data that can triangulate, contextualize, and clarify our quantitative findings.

This year, we significantly expanded the role of qualitative data in our research design by interviewing development professionals on a continuous basis from July 2024–July 2025. In addition to using these insights to clarify findings from our survey design, we also used qualitative data to generate new hypotheses to test as part of our survey, especially as related to emerging practices in AI-assisted and AI-driven development paradigms.

The interview guides used throughout this process were designed to touch on a range of foundational topics in the domain of AI use in software development, while affording flexibility to probe topics of interest raised by participants as they emerged. Interview sessions were designed to last approximately 90 minutes each, and were conducted remotely.

In total, we interviewed 78 participants who were confirmed to be professionally involved in software development through a screener survey and phone screen. All interviews were video- and audio-recorded. All interviews were transcribed using automated software. Quotations appearing in the final publication of this report were revisited and transcribed manually prior to inclusion. Words added or altered in participant quotations by the authors of this report are indicated by brackets ([]) and words removed are indicated by ellipses (...). Edits were made only in cases where required for clarity or anonymity.

Statistical analysis

This section outlines our analytical method, which is heavily steeped in the works of *Statistical Rethinking* and *Regression and Other Stories*.[1,2] We could attach a footnote from one of those to each sentence. We walk through our entire workflow with a simplified example so you can determine the appropriate level of trust to put into our results and replicate our approach. The 2024 DORA Report explored the theoretical principles behind our analysis in the Methodology section. This year, we focus on the practical application: how we conducted our analysis.

To do so, we are going to go through a simulated, simplified, and idealized example. We're going to smooth over or even skip some of the complexity. This guide is a high-level overview; some aspects of a full analysis are discussed lightly or not at all to maintain focus on the core workflow. Here are the core variables in this toy analysis:

1. `individual_experience` = how much experience someone has in this type of role (manifest variable)
2. `resources` = resources (for example, tools) available to do work (manifest variable)
3. `stability` = does the organization have stable priorities (manifest variable)
4. `individual_effectiveness_score` = a factor made up of four indicators
5. `ai_adoption` = does the organization have stable priorities

If you finish this little section, hopefully you have a better sense of whether to trust us, a desire to replicate this work, and a couple of new statistics tricks.

Step 1: Defining our causal theory (the DAG)

All statistical models contain causal assumptions; our approach is to make them explicit. Correlation may not imply causation, but how you think about causation will impact your correlations. We codify our theory about how the world works in a directed acyclic graph (DAG).[3]

This DAG is our map of the causal landscape, built from prior research, qualitative work, and domain expertise. By visualizing our assumptions, we make them transparent and debatable, which is a cornerstone of rigorous science.[4]

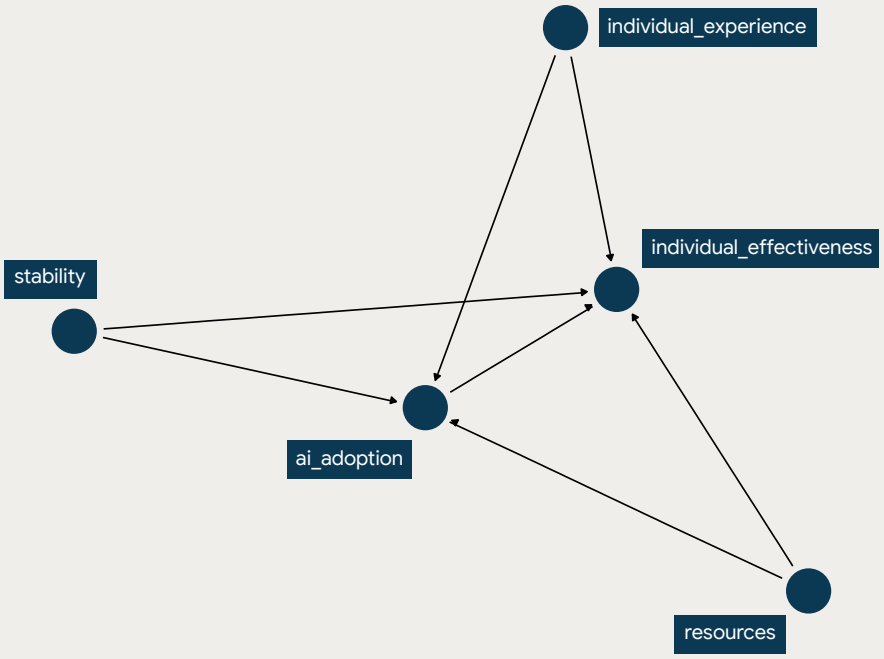
For this example, our extremely simplified DAG hypothesizes that `individual_experience`, `resources`,[5] and `stability` are common causes of both `ai_adoption` and `individual_effectiveness`.

```
# Load necessary libraries
library(dagitty)
library(ggdag)
library(lavaan)
library(tidyverse)
library(brms)
library(tidybayes)
library(ggplot2)
library(emmeans)
library(performance)
library(semPlot)


# Define the causal relationships in our simplified DAG
simple_dag <- dagitty('dag {
 ai_adoption [exposure]
 individual_effectiveness [outcome]
 individual_experience [adjusted]
 resources [adjusted]
 stability [adjusted]


 individual_experience -> ai_adoption; resources -> ai_adoption;
stability -> ai_adoption
 ai_adoption -> individual_effectiveness
 individual_experience -> individual_effectiveness; resources ->
individual_effectiveness; stability -> individual_effectiveness
}')


# Plot the DAG to visualize our theory
ggdag(simple_dag, text = FALSE, use_labels = "name") + theme_dag_
blank()
```



This output diagram visualizes our theory. Our focus is `ai_adoption`'s impact on `individual_effectiveness`, but to properly estimate that, we need to understand the broader context. The DAG identifies several key confounders—factors that influence both our cause and our effect. These are:

`individual_experience` (with role)

`resources` (required to do role are available)

`stability` (priority stability)

We'll let the DAG explicitly identify them later.

Generating fake data from our causal theory

Let's use the causal structure defined in our DAG to simulate a dataset for this example. This is fake data.

Creating data from a known causal structure gives us a perfect "ground truth." This is incredibly helpful because it allows us to stress-test our methods. If our analysis can't recover the correct answer from this clean, perfectly known data, it certainly can't be trusted with messy, real-world data.

However, this is also a simplification. Real data contains complexities like non-linear relationships, measurement error, unknown causal structures, and missing values that we exclude from this simplified example.

```
# Define the model with specific path coefficients to generate data
model_specification <- "
 # -- Measurement Model --
 ai_adoption_factor =~ 0.8*use + 0.7*reliance + 0.6*trust
 individual_effectiveness_factor =~ 0.8*effective +
0.7*productivity + 0.7*impactful + 0.6*flow


 # -- Structural Model (Paths based on the DAG) --
 ai_adoption_factor ~ 0.4*individual_experience + 0.3*resources +
0.2*stability
 individual_effectiveness_factor ~ 0.3*ai_adoption_factor +
0.3*individual_experience + 0.2*resources + 0.15*stability"
set.seed(2025)
simulated_data <- simulateData(model_specification, sample.nobs =
500)


#get a sense of the data
summary(simulated_data)
glimpse(simulated_data)
```

Step 2: Evaluating the measurement model (CFA)

Before testing our causal or structural theory, we must ensure our measurement tools are sound using a confirmatory factor analysis (CFA).[6] This step validates that our survey items reliably measure their intended latent constructs (Kline, 2015). We evaluate the model by checking for good global fit, strong local factor loadings, and no major points of strain via modification indices.

```
# Define the measurement-only portion of our model
measurement_model <- "
 ai_adoption_factor =~ use + reliance + trust
 individual_effectiveness_factor =~ effective + productivity +
impactful + flow
"
# Fit the CFA model to our simulated data
cfa_fit <- cfa(measurement_model, data = simulated_data)

# Review the fit indices, loadings, and modification indices
summary(cfa_fit, fit.measures = TRUE, standardized = TRUE)
modificationindices(cfa_fit, sort = TRUE, minimum.value = 10)
```

This is some of the output for that model.

```
> summary(cfa_fit, fit.measures = TRUE,
standardized = TRUE)
lavaan 0.6-19 ended normally after 34 iterations

 Estimator                         ML
 Optimization method           NLMINB
 Number of model parameters        15

 Number of observations           500

Model Test User Model:

 Test statistic                 7.579
 Degrees of freedom                13
 P-value (Chi-square)           0.870

Model Test Baseline Model:
 Test statistic               651.579
 Degrees of freedom                21
 P-value                    0.000

User Model versus Baseline Model:

 Comparative Fit Index (CFI)    1.000
 Tucker-Lewis Index (TLI)       1.014
```

```
Loglikelihood and Information Criteria:

 Loglikelihood user model (H0)       -5515.401
 Loglikelihood unrestricted model (H1)
-5511.612

 Akaike (AIC)                11060.803
 Bayesian (BIC)               11124.022
 Sample-size adjusted Bayesian (SABIC)
11076.411

Root Mean Square Error of Approximation:

 RMSEA                          0.000
 90 Percent confidence interval - lower     0.000
 90 Percent confidence interval - upper     0.023
 P-value H_0: RMSEA <= 0.050          0.999
 P-value H_0: RMSEA >= 0.080          0.000

Standardized Root Mean Square Residual:

 SRMR                           0.017
```

How to interpret the CFA output

The lavaan[7] output provides a rich set of diagnostics. Here's how to interpret the key sections using your results.

1. The chi-square test (x2). This is the test of perfect fit. It tests the null hypothesis that the model fits the data exactly.

Your result: Test statistic = 7.579, df = 13, P-value = 0.870

Guideline: We hope for a non-significant result (p > .05).

Interpretation: The p-value is very high (0.870), meaning we cannot reject the null hypothesis. The test indicates that our model's structure is statistically indistinguishable from a perfect fit.[8]

Warning: This is the traditional, formal statistical test of model fit. However, with very large samples like DORA's, this test is overly sensitive and almost always indicates a "poor fit" even when the model is excellent. Because of this, we use it as a reference but rely more heavily on the practical indices below.

2. Incremental fit indices (CFI & TLI). These indices compare our model's fit to a "worst-case" baseline model where no variables are related.

Your result:
CFI = 1.000, TLI = 1.014

Guideline: We check for values > 0.90 (acceptable) and > 0.95 (excellent).

Interpretation: The values are at or above the theoretical maximum of 1.0. This signals a perfect fit relative to the baseline model. (Note: TLI can sometimes exceed 1.0 in well-fitting models.)

3. Absolute error indices (RMSEA & SRMR). These indices measure the "badness-of-fit" or the average error between the model's predictions and the actual data.

Your RMSEA result:
RMSEA = 0.000 with a 90% CI of [0.000, 0.023]

Guideline: We check for a point estimate < 0.08 (acceptable) and < 0.06 (excellent).

Interpretation:
Your RMSEA is zero, with the entire confidence interval falling well below the threshold for excellent fit. This suggests there is virtually no error of approximation.

Your SRMR result:
SRMR = 0.017

Guideline:
We check for a value < 0.08.

Interpretation: The value is extremely low, indicating that the average standardized difference between the observed and predicted correlations is tiny.

4. Factor loadings and modification indices (output not shown). Use the standardized factor loadings to inspect if the loadings are substantial and similar from within the factor.[9] Use the modification indices to locate areas of local strain in your model. High modification indices indicate areas of potential misspecification.

In this example, across the board, every single fit index points to a model that fits the data near-perfectly. This is exactly what we hope to see in this specific scenario. Because we generated the data directly from this model's specifications, a perfect fit confirms that our analytical method is working correctly and can recover a known structure. However, it's crucial to note that seeing results this flawless with real-world survey data would be highly unusual and almost worrisome. Such a perfect fit in a real analysis might suggest that the model is too simple for the data's complexity or, in some cases, that the model has been "overfit" through extensive tweaking.

Step 3: Testing and evaluating the model structure (SEM)

With validated measures, we use structural equation modeling (SEM) to perform a holistic fit test of our entire causal theory. The purpose of SEM is to compare the covariance structure implied by our DAG to the covariance structure observed in our data.

A good model is one where the theory's predictions closely match reality. We are evaluating how well our theoretical model aligns with the observations.

```
#Define our theoretical model


# Define the model with specific path
coefficients to generate data

structural_model <- "

 # -- Measurement Model --

 ai_adoption_factor =~ use + reliance + trust

 individual_effectiveness_factor =~ effective +
productivity + impactful + flow


 # -- Structural Model (Paths based on the DAG)
--

 ai_adoption_factor ~ individual_experience +
resources + stability

 individual_effectiveness_factor ~ ai_adoption_
factor + individual_experience + resources +
stability

"



# Fit the full SEM to the data to estimate the
path coefficients

sem_fit <- sem(model_specification, data =
simulated_data)



# Run a full summary to get fit measures,
loadings, and path coefficients

sem_summary <- summary(sem_fit, standardized =
TRUE, rsquare = TRUE)

sem_summary
```

```
# Inspect specific areas of local fit

residuals(sem_fit, type = "standardized")

modificationindices(sem_fit, sort. = TRUE,
minimum.value = 10)


# --- Generate a Minimal, Monochrome SEM Path
Diagram ---

semPaths(sem_fit,

     # --- Core Content ---

     what = "std",      # IMPORTANT: Show
standardized coefficients on paths

     whatLabels = "est",   # Show the estimate
value (the coefficient)


     # --- Layout & Sizing ---

     layout = "tree2",    # A clean hierarchical
layout

     rotation = 2,      # Rotates the layout for
better viewing (often top-to-bottom)

     sizeMan = 8,      # Size of manifest
variables (squares)

     sizeLat = 10,      # Size of latent
variables (circles)

     edge.label.cex = 0.8,  # Font size for the
path coefficients

     nCharNodes = 0,     # Ensures full variable
names are displayed


     # --- Minimalist Aesthetics ---

     style = "ram",     # A clean drawing style
from the RAM specification

     residuals = FALSE,   # HIDE residual
variances to reduce clutter

     intercepts = FALSE,   # HIDE intercepts for
simplicity


     # --- Monochrome Theme ---

     color = "white",    # Fill color for all
shapes

     edge.color = "black",  # Color for all
arrows

     border.color = "black", # Border color for
shapes

     border.width = 1.5,   # Thickness of the
borders

     label.color = "black"  # Color of the text
on the arrows

)
```

How we evaluate the SEM

Judging if the model is "good" requires a holistic check of several indicators. We group these into two categories:

1. Global fit: The 30,000-foot view.[9] These indices tell you how well the model as a whole reproduces the data.

The chi-square test (x2): This is the test of perfect fit. It tests the null hypothesis that the model fits the data exactly.

CFI & TLI: The comparative fit index and Tucker-Lewis index measure how much better our model is than a "worst-case" model. We check for values > 0.90 (acceptable) or > 0.95 (excellent).

RMSEA: The root mean square error of approximation is a "badness-of-fit" index measuring the average model error. We check for values < 0.08 (acceptable) or < 0.06 (excellent).

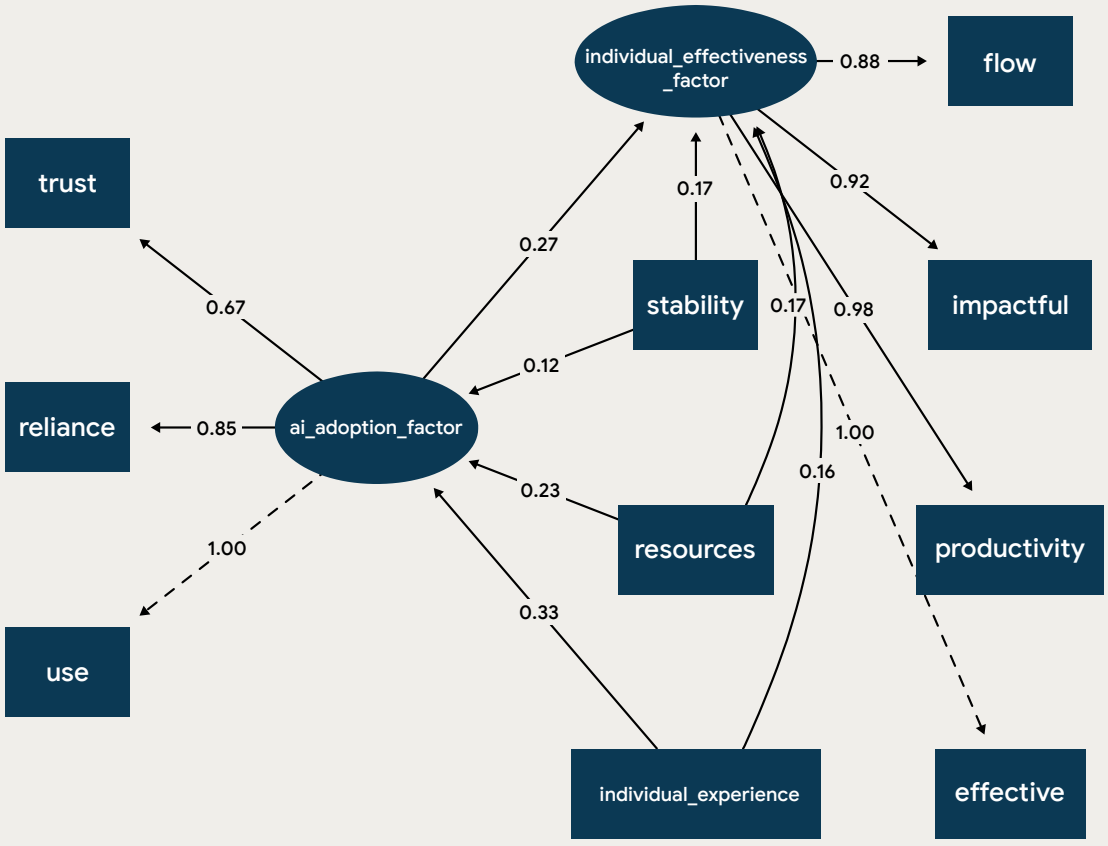
2. Local fit: Checking under the hood. These diagnostics help us find specific points of strain within the model.

Factor loadings: In the standardized output, we check for high, significant loadings (ideally > 0.50) for the items on their respective latent factors. This confirms our measurement model is strong.

Standardized residuals: These show the error for each individual relationship in the model. We check for an absence of large values (for example, no absolute values > 2.58), which would indicate a specific relationship is poorly predicted by our theory.

Modification indices: These "what-if" statistics tell us where the model is under the most strain. We check for large values to diagnose problems (like a survey item measuring two concepts at once), but we avoid blindly modifying our model based on them without strong theoretical justification.

A good model is one that has acceptable global fit, strong factor loadings, and no major signs of local strain. It's this constellation of evidence that gives us confidence in the model's structure.



Testing the theoretical model's implications

Our DAG also generates a set of testable predictions known as "implied conditional independencies." These are focused on the circumstances in which these variables are not related. Think of this as testing a theoretical model's alibis. If they hold up, you have more reason to believe it.

When we run `impliedConditionalIndependencies(simple_dag)', we notice:

```
ind_ _||_ rsrc
ind_ _||_ stbl
rsrc _||_ stbl
```

These statements are a core prediction of our causal model, stating that a person's individual experience, their available resources, and their team's stability are all independent of one another. In other words, knowing the level of available resources for a team provides no information about an individual's experience on that team or the team's overall stability.

We first check this informally by examining the covariances in the original model. They're all near 0, so we have some pretty good evidence.

We can also try a more formal test, where we build these implications in our structural equation model. We would do this by constraining those parameters to 0. In our last model, we asked the model to estimate those parameters, as we believed they were free to be whatever they wanted to be. Now, we're saying they're zero.

The key result comes from the anova() function, which performs a formal Chi-squared difference test. This provides a formal comparison between your original model (sem_fit) and the more restrictive model (constrained_fit) where you forced the covariances to be zero. The key question is: "Did forcing those relationships to be zero significantly harm the model's fit?"

```
# check in the "Covariances" section of this
output

summary(sem_fit, standardized = TRUE)

# --- Formally Test the Implied Conditional
Independencies ---

#get the implied conditional independencies

impliedConditionalIndependencies(simple_dag)


#we can just inspect the covariances

sem_summary$pe %>% filter(op == "~~",

             lhs %in% c("individual_experience",

                   "resources",

                   "stability"))


# 1. Define a new, constrained model string

constrained_model_string <- "

 # -- Measurement Model --

 ai_adoption_factor =~ use + reliance + trust

 individual_effectiveness_factor =~ effective +
productivity + impactful + flow


 # -- Structural Model (Paths based on the DAG)
--

 ai_adoption_factor ~ individual_experience +
resources + stability

 individual_effectiveness_factor ~ ai_adoption_
factor + individual_experience + resources +
stability

 # --- NEW CONSTRAINTS ---

 # We now explicitly force the covariances to
zero based on the DAG's implication

 individual_experience ~~ 0*resources

 individual_experience ~~ 0*stability

 resources ~~ 0*stability

"


# 2. Fit the constrained model

constrained_fit <- sem(constrained_model_string,
data = simulated_data)


# 3. Compare the original model (sem_fit) to the
constrained model

# A non-significant p-value means the constraints
are supported by the data.

anova(sem_fit, constrained_fit)
```

```
                Df  AIC  BIC Chisq Chisq diff RMSEA Df diff Pr(>Chisq)
sem_fit         28 10952 11040 21.454
constrained_fit 31 15319 15420 24.980   3.5267 0.018739    3   0.3173
```

The results suggest that restricting or constraining those three covariances to zero did not significantly hurt the model.

Chisq diff (Chi-squared difference):
This is the raw measure of how much worse the fit got when you added the constraints.

Your result:
3.5267. By itself, this number is hard to interpret.

Df diff (Difference in degrees of freedom):
This shows how many constraints you added.

Your result:
3. This is correct, as you forced three covariances to zero.

Pr(>Chisq) (The P-value):
This is the most important number. It tells you the probability of seeing a Chisq diff of 3.5267 or larger if the constraints were actually true in the population.

Your result:
0.3173.

In short, the implications held up and the constrained model is the more parsimonious approach.

Step 4: Estimating comparisons with a Bayesian model

After gaining confidence in our DAG's structure, we turn to our main question. We use our DAG to identify the correct adjustment set and then fit a Bayesian model. While our framework is causal, we interpret our results as principled comparisons, acknowledging the limitations of observational data (Gelman et al., 2020).

Setting priors

Our approach of testing a predictor across numerous outcomes creates a multiple comparisons challenge. To address this, we implement a consistent, skeptical Bayesian prior for our predictor in every model.[10] This acts as a principled defense against being fooled by randomness. By setting a skeptical prior (student_t(3, 0, 0.5)),[11] we formally build in the belief that the true effect is likely small, which tames the false positive rate.[12]

Determining the adjustment set

DAG will let you know what variables to include, and perhaps more importantly, what variables not to include, in your model. If your DAG is correct, the inclusion of these variables should prevent biasing pathways from biasing your estimate of AI adoptions (exposure) impact on individual effectiveness (outcome).

```
# Identify adjustment set

adjustmentSets(simple_dag, exposure = "ai_
adoption", outcome = "individual_effectiveness")


# Prepare data

model_data <- simulated_data %>%

 mutate(

  ai_adoption_score = rowMeans(select(., use,
reliance, trust)),

  individual_effectiveness_score =
rowMeans(select(., effective, productivity,
impactful, flow))

 ) %>%

 mutate(across(c(ai_adoption_score, individual_
experience, resources, stability), scale))


# Define priors

priors <- c(prior(student_t(3, 0, 0.5), class
= "b"), prior(student_t(3, 0, 2.5), class =
"Intercept"), prior(exponential(1), class =
"sigma"))
```

```
# Fit models

baseline_model <- brm(formula = individual_
effectiveness_score ~ individual_experience +
resources + stability, data = model_data, family
= gaussian(), prior = priors, chains = 4, iter
= 4000, warmup = 2000, seed = 2025, silent = 2,
refresh = 0)

full_model <- brm(formula = individual_
effectiveness_score ~ ai_adoption_score +
individual_experience + resources + stability,
data = model_data, family = gaussian(), prior =
priors, chains = 4, iter = 4000, warmup = 2000,
seed = 2025, silent = 2, refresh = 0)


# Compare models with LOO

# loo_compare <- loo(baseline_model, full_model)

# print(loo_compare)


#coefficient summaries with 89% credibility
intervals

posterior_summary(full_model, probs = c(1-
.11/2,.11/2))


#type s error (approximately a pvalue)

1 - mean(as.matrix(full_model)[,"b_ai_adoption_
score"] >0)
```

We believe a single metric is never enough to understand a model. Instead, we evaluate the evidence through four key lenses, embracing the uncertainty in our data as a source of knowledge.

1. Does predictive accuracy improve? (The LOO result)

Yes. The LOO comparison shows a clear preference for the full_model (elpd_diff = 10.7, se_diff = 5.2).[13] This is our first piece of evidence: including ai_adoption_score creates a model that is expected to make better predictions on new data. It's not just statistical noise; it's a predictively useful variable.

2. Is the effect consistently on one side of zero? (Type S error)

Yes. Checking at the posterior summary for b_ai_adoption_score:

```
                Estimate Est.Error  Q94.5   Q5.5
b_ai_adoption_score  0.199   0.041  0.265   0.134
```

The 89% credible interval (from Q5.5 to Q94.5) for the AI adoption coefficient is [0.13, 0.26]. The entire interval is well above zero. This means there is a very low probability of a Type S (Sign) error. We can be highly confident that the relationship is positive.

3. Do we have a good sense of the effect's size? (Type M Error)

Yes. The model provides a clear sense of the magnitude. Our best estimate is a 0.20 standard deviation increase in effectiveness for every one standard deviation increase in AI adoption. The credible interval [0.13, 0.26] gives us a plausible range for this effect. It's not a huge effect, but it's not trivial either, and the estimate is reasonably precise. This clarity helps us avoid a Type M (Magnitude) error by preventing us from overstating the effect's importance.

4. Does this align with our theory?

Yes. Our initial DAG hypothesized a direct, positive causal path from AI adoption to individual effectiveness. All the statistical evidence we've gathered—from the SEM's structural validation to the LOO comparison and the final posterior estimate—is consistent with this theoretical claim. The data supports the story we started with.

Synthesis

By combining these four perspectives, we move beyond a simple "Is it significant?" mindset. The evidence converges to show that the relationship between AI adoption and effectiveness is predictively useful, directionally stable, modest but clearly estimated in magnitude, and theoretically sound. This approach helps us avoid "getting jerked around by noise patterns that happen to exceed the statistical significance threshold."[14]

Step 5: Diagnosing the model for trustworthiness[15]

Fitting a model is not the end. We perform a series of rigorous diagnostic checks to ensure the results are reliable.[16] This includes checking for MCMC convergence (hatR < 1.01, high ESS), running posterior predictive checks to ensure the model can reproduce the observed data, and checking model assumptions via residual analysis (Gelman et al., 2013). This topic is worth a chapter in itself. We are simply going to share some important diagnostic checks to consider. These are worth knowing because their applicability is widespread, from basic models to advanced ones. The basic categories are:

Computational health:

We first ensure the model's algorithm ran correctly and produced stable estimates. This is a technical check for the model's computational engine, ensuring its results are reliable (for example, checking that the R^ statistic is less than 1.01).

Predictive alignment:

We then check if the model's predictions align with the real-world data we started with. We use the model to simulate data and see if it "checks like" the data we actually observed. A model that can't recreate the past can't be trusted to explain the present.

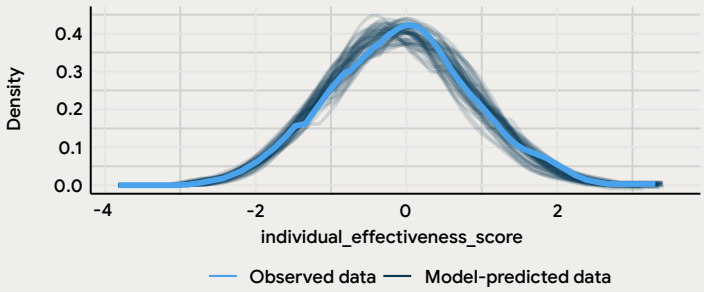
Statistical validity:

Finally, we verify that the model's core statistical assumptions were met. This involves inspecting the model's errors (its residuals) to ensure we haven't violated fundamental principles, such as the assumption of a linear relationship.

```
print("--- Convergence Diagnostics (R-hat, ESS) ---")
summary(full_model)
print("--- Visual Trace Plots ---")
plot(full_model, N = 4, ask = FALSE)
print("--- Posterior Predictive Check ---")
pp_check(full_model, ndraws = 100)
print("--- Residual Assumption Checks ---")
check_model(full_model)
```

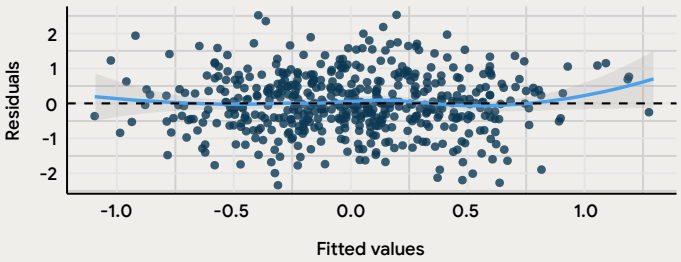
Posterior predictive check

Model-predicted lines should resemble observed data line



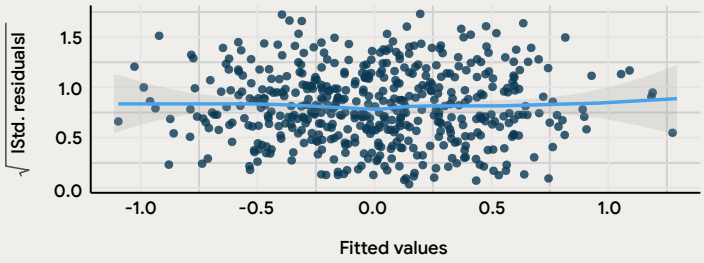
Linearity

Reference line should be flat and horizontal



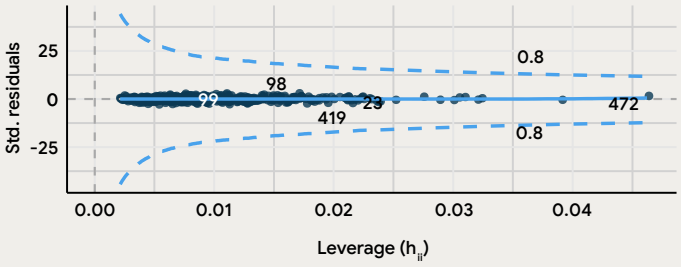
Homogeneity of variance

Reference line should be flat and horizontal



Influential observations

Points should be inside the contour lines



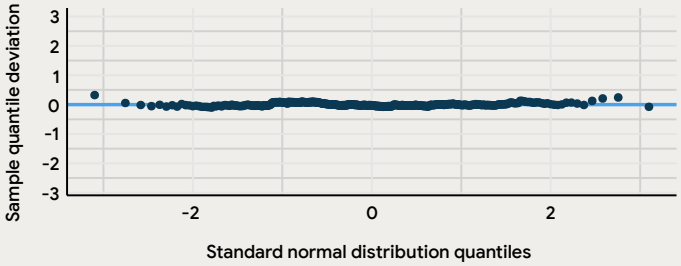
Collinearity

High collinearity (VIF) may inflate parameter uncertainty



Normality of residuals

Dots should fall along the line

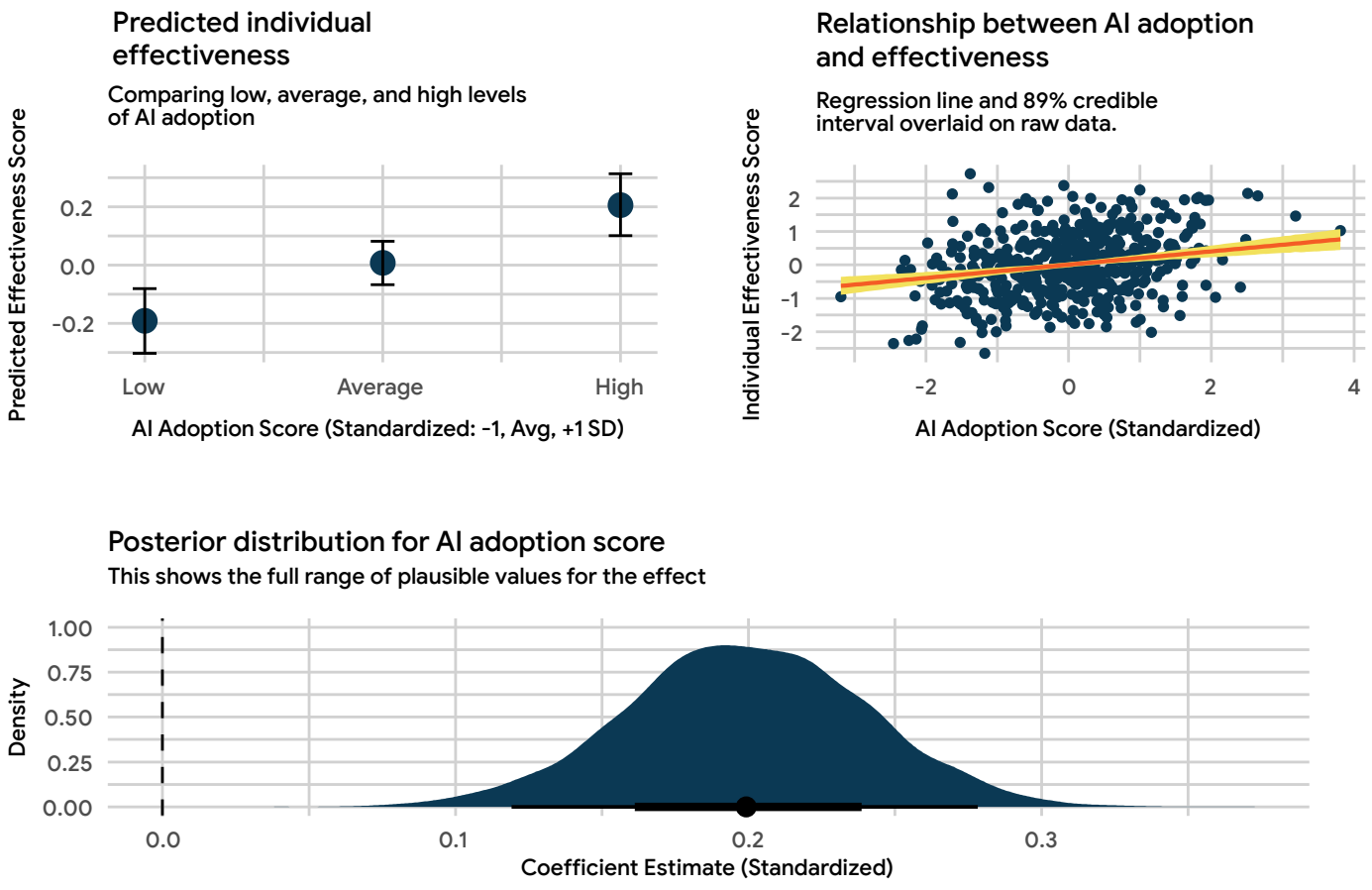


Step 6: Visualizing the estimated effect

Finally, we translate our statistical results into intuitive visualizations. We interpret our model by examining the full posterior distribution of our main parameter, plotting conditional predictions (or estimated marginal means) to understand the magnitude of the comparison, and viewing the regression line in the context of the raw data.

For our final results, we report 89% credible intervals, a choice that deliberately shows the arbitrariness of p-value-centric thinking and focuses on the stable, high-density region of the posterior (McElreath, 2020).

Three views of the effect of AI adoption on individual effectiveness



Top: Predictions on the outcome scale. Bottom: Posterior distribution of the standardized coefficient.

```
# --- STEP 6: Visualizing the Estimated Effect
---
print("Generating visualizations...")


# 6a. The Posterior Distribution
plot_posterior <- full_model %>%
  spread_draws(b_ai_adoption_score) %>%
  ggplot(aes(x = b_ai_adoption_score)) +
  stat_halfeye(fill = "#1565C0") +
  geom_vline(xintercept = 0, linetype = "dashed")
+
  labs(
    title = "Posterior Distribution for AI
Adoption Score",
    subtitle = "This shows the full range of
plausible values for the effect.",
    x = "Coefficient Estimate (Standardized)",
    y = "Density"
  ) +
  theme_minimal()


# 6b. The Comparison Plot (Estimated Marginal
Means)
emm_results <- emmeans(full_model,
                       specs = ~ ai_adoption_
score,
                       at = list(ai_adoption_
score = c(-1, 0, 1)),
                       prob = 0.89) # 89%
Credible Interval


plot_comparison <- as.data.frame(emm_results) %>%
  ggplot(aes(x = ai_adoption_score, y = emmean))
+
  geom_point(size = 4, color = "#1565C0") +
  geom_errorbar(aes(ymin = lower.HPD, ymax =
upper.HPD), width = 0.1, linewidth = 1) +
  labs(
    title = "Predicted Individual Effectiveness",
    subtitle = "Comparing low, average, and high
levels of AI adoption.",
    x = "AI Adoption Score (Standardized: -1,
Avg, +1 SD)",
    y = "Predicted Effectiveness Score"
  ) +
  theme_minimal() +
  scale_x_continuous(breaks = c(-1, 0, 1), labels
= c("Low", "Average", "High"))


# --- 6c. The Relationship in Context
(Scatterplot with Regression Line) ---


# Create a reference grid: a sequence of points
along the range of our predictor
plot_grid <- ref_grid(full_model,
                      at = list(ai_adoption_score
= seq(min(model_data$ai_adoption_score),


max(model_data$ai_adoption_score),


length.out = 100)))


# Get the predictions (estimated marginal means)
at each point in our grid
emm_plot_data <- emmeans(plot_grid, "ai_adoption_
score", prob = 0.89) %>% as.data.frame()


# Now, create the plot with this new, smooth data
plot_relationship <- ggplot(emm_plot_data, aes(x
= ai_adoption_score, y = emmean)) +
  geom_point(data = model_data, aes(y =
individual_effectiveness_score), alpha = 0.2,
color = "gray50") +
  geom_line(color = "#1565C0", linewidth = 1.5) +
# No group aesthetic needed now
  geom_ribbon(aes(ymin = lower.HPD, ymax = upper.
HPD), alpha = 0.2, fill = "#1565C0") +
  labs(
    title = "Relationship Between AI Adoption and
Effectiveness",
    subtitle = "Regression line and 89% credible
interval overlaid on raw data.",
    x = "AI Adoption Score (Standardized)",
    y = "Individual Effectiveness Score"
  ) +
  theme_minimal()
```

```
# --- Print the corrected plot ---
print(plot_relationship)

# --- Print the final plots to the screen ---
print(plot_posterior)
print(plot_comparison)
print(plot_relationship)



# --- Combine the three plots using patchwork ---
# The '+' operator puts plots side-by-side
# The '/' operator stacks them vertically
combined_plot <- (plot_comparison + plot_
relationship) /
  plot_posterior +
  plot_annotation(
    title = "Three Views of the Effect of AI
Adoption on Individual Effectiveness",
    caption = "Top: Predictions on the outcome
scale. Bottom: Posterior distribution of the
standardized coefficient."
  )
```

```
# --- Print and save  the final combined plot ---
print(combined_plot)

ggsave(
  combined_plot,
  filename = "combined_plot.svg",
  height = 6 * .75,
  width = 9 *.75,
  dpi = 600
)



print("--- Workflow Complete ---")
```



Conclusion

This workflow—from explicit theory to rigorous validation, estimation, and diagnostics—is designed to produce transparent, trustworthy, and robust insights.

We're hoping that sharing our process here and the Our research model and its theory chapter puts readers in a position to better evaluate our work. We're also hopeful that this makes replication possible and encourages people to leverage some fun statistical approaches.



1. McElreath, Richard. *Statistical Rethinking: A Bayesian Course with Examples in R and STAN* (Chapman and Hall/CRC, 2020).
2. Gelman, Andrew, Jennifer Hill, and Aki Vehtari. Regression and Other Stories of Analytical Methods for Social Research (Cambridge University Press, 2020).
3. The Our research model and its theory chapter provides an overview of this codified theory.
4. Pearl, Judea. *Causality* (Cambridge University Press, 2009).
5. The Our research model and its theory chapter has a much more comprehensive overview.
6. Before we do a confirmatory factor analysis, we do an exploratory factor analysis to understand how the factors fall out before we constrain parameters with a theoretical model. This, in combination with other diagnostics, helps us better understand where our theory might have areas of poor fit.
7. Rosseel Y (2012). *"lavaan: An R Package for Structural Equation Modeling."* Journal of Statistical Software, 48(2), 1–36. https://doi.org/10.18637/jss.v048.i02
8. This is good news for the simpler model. It is essentially saying we don't lose a lot of information by not freely estimating certain parameters.
9. There are formal tests you could use to evaluate this.
10. Good modeling isn't just about fit; it's about finding the simplest model that fits well (Occam's razor). You could keep adding parameters to achieve this. The questions are which parameter is justifiable and how much can you trim without losing much understanding.
11. We also test for how robust the results are to various priors. In the end, we've decided we want to prevent false positives more than false negatives. We don't want people exploring certain areas based on us being tricked by noise.
12. There is a good discussion on the topic here: https://discourse.mc-stan.org/t/why-studentt-3-0-1-for-prior/8102
13. For interaction terms, we use an even more skeptical prior (normal(0, 0.3)), as these effects are expected to be smaller.
14. The model on top is the model that is making the best predictions. The elpd_diff should be greater than the se_diff. I start feeling especially confident when elpd_diff is greater than 2x the se_diff (so, greater than 10.4 or 5.2 x 2).
15. The earlier you start evaluating these assumptions, the better.
16. Exploring the distributions and patterns in descriptive stats is a sometimes tedious, but crucial part that takes place at the start, before models are even being considered.

Our research model and its theory

"Theory represents an essential decision that causes the world to appear wholly different—in a wholly different light. Theory is a primary, primordial decision, which determines what counts and what does not ..."

Byung-Chul Han[1]

Derek DeBellis

Quantitative User Experience Researcher, Google Cloud

This chapter outlines the theoretical model that underpins our analysis and estimates. It is a product of discussion with the DORA community, the experiences of subject-matter experts who are responsible for implementing changes across the company, literature, and troves of qualitative data. The model isn't just an exercise of connecting boxes with lines. It is critical because the theoretical and unavoidably casual assumptions contained within it guide the analysis. Small changes have large implications on the analysis. Correlation does not imply causation, but our assumptions about causation do impact the correlations we find.

This model is unique for DORA in two key ways. First, we are primarily focused on the impact of AI adoption and the conditions under which this impact is modified (AI capabilities). This means the model is primarily designed so we can get accurate estimates about the effect of AI adoption on the outcomes we believe are important to technology-driven organizations.

Normally, the model is trying to predict the relationship between many capabilities and many outcomes.

Second,[2] the model is evaluated at the level of a structural equation model, but the results of that analysis are only to establish the model's plausibility. From there, we build targeted Bayesian models to generate more focused and granular estimates.

The Methodology chapter goes into considerable detail about this.

The basic flow of this chapter is as follows:

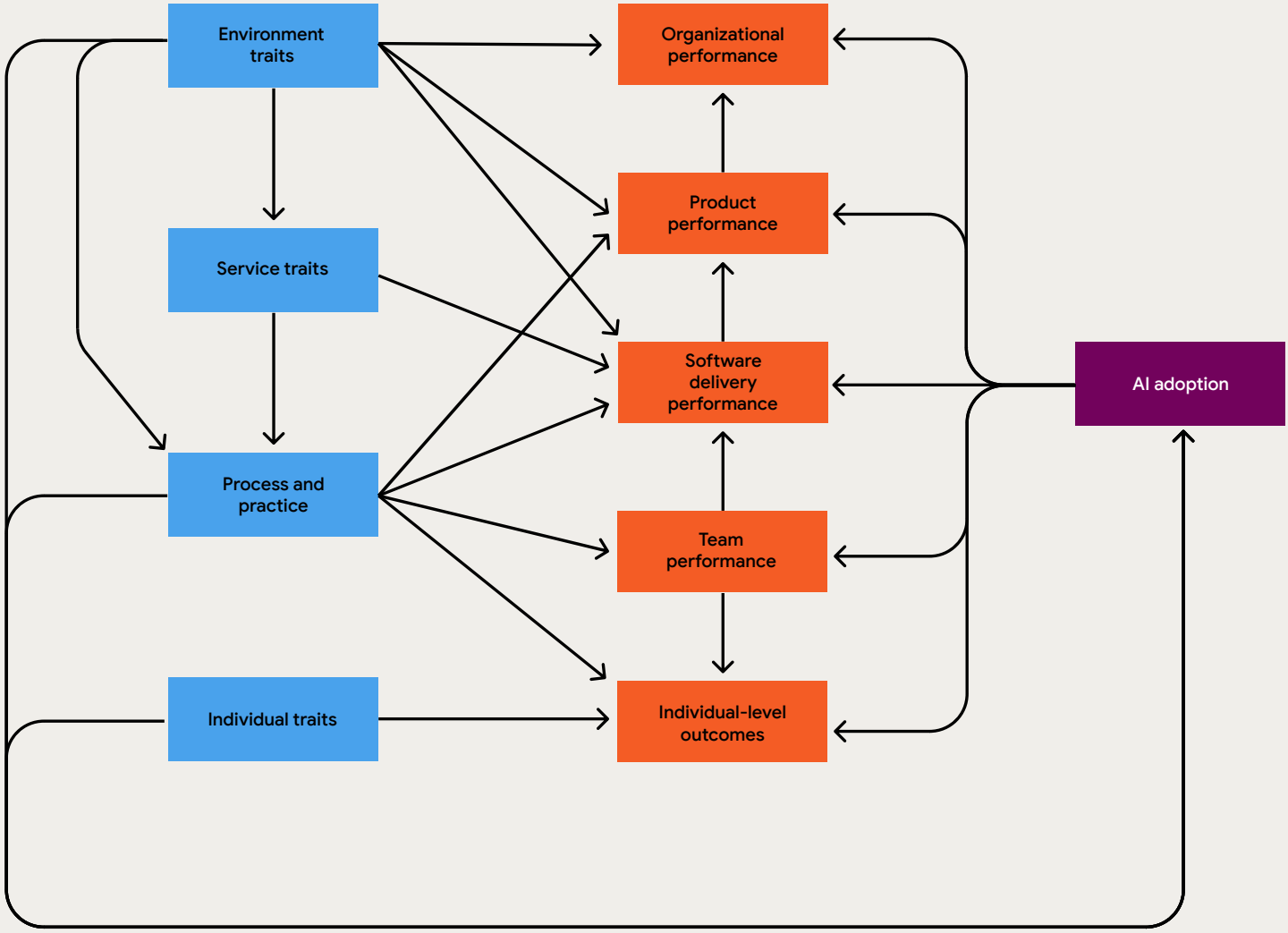
The model: Introduction to our overall model

The concepts: Description of the high-level concepts of the model

The theory: Outline of the theoretical justification for each pathway

This chapter, combined with the Methodology chapter, provides an overview of both the theory and analysis that are behind the results. It should lay bare our assumptions. We hope this understanding provides enough information to replicate, leverage, and evaluate our work.

The model



The concepts (represented as boxes) in our causal model are high-level groupings. These groupings simplify the visualization and don't necessarily represent the exact measurements used in our analysis. The model contains contextual concepts that help us understand the respondents' circumstances. This includes environment traits, service traits, processes, and individual traits.

AI adoption is a particular latent factor from our confirmatory factor analysis that is a composite of reflexive use, trust, and reliance. Then there are our outcomes, which are explained in more detail in the executive summary and the AI impacts chapter.

We like to think that no concept here is superfluous. Getting good estimates of AI adoption's effect requires mapping the tangled reality in which that effect resides.

We use structural equation modeling and directed acyclic graphs to evaluate how well this theoretical model aligns with observed data. When the model is verified, we use the DAG to adjust our analyses to get better estimates.

The Methodology chapter dives into the minutiae. Here, we're going to focus on the concepts and theory undergirding this model.

The concepts

AI adoption

AI adoption measures the integration of AI into an individual's workflow and mindset. This concept distinguishes simple tool usage from a deeper partnership with technology.

We measured AI adoption through three core indicators:

- Trust
- Reflexive use
- Reliance

Processes and practices

This category captures a wide array of capabilities. Some are AI-specific. Some are group-level processes. Some are individual-level processes. They're all representative of actions and ways of working.

There are many constructs associated with this:[3]

- Clear and communicated AI stance
- Healthy data ecosystems
- AI-accessible internal data
- Strong version control practices
- Working in small batches
- User-centric focus
- Quality internal platforms

Each of these constructs is part of our inaugural AI capabilities model discussed in the AI Capabilities Model chapter.

Individual traits

This category captures the specific characteristics of an individual, including their role, age, and tenure on a team. It also covers the nature of their work, such as the amount of time they spend on AI-related tasks. These details provide crucial context for understanding a person's experience and how they interact with technology.

This concept is explored or constituted by the following observations:

- Time spent using AI
- Years spent on team
- Individual role
- Individual age
- Individual tasks
- AI-specific tasks

Environmental and organizational traits

These concepts describe the broader structural context in which work happens. This includes stable factors like company size and industry, as well as more dynamic conditions like the availability of resources and the stability of priorities. This environment creates the conditions that either enable or constrain technology adoption and overall performance.

Service traits

Service traits define the key characteristics of the primary application or service on which an individual works. Understanding a service's age, criticality, and whether it is AI-infused is essential for contextualizing performance metrics and the relevance of certain technical practices.

Organizational performance

This is a high-level measure of the overall success of the organization, based on characteristics like profitability, market share, and customer satisfaction.

Team performance

This factor measures the perceived effectiveness and collaborative strength of an individual's immediate team.

Product performance

This factor measures the success and quality of the products or services the team is building, based on characteristics such as helping users accomplish important tasks, keeping information safe, and performance metrics like latency.

Software delivery performance

- **Software delivery throughput:** This represents the speed and efficiency of the software delivery process. See the Understanding your software delivery performance chapter for more details.
- **Software delivery instability:** This captures the quality and reliability of the software delivery process. See the Understanding your software delivery performance chapter for more details.

Individual outcomes

- **Code quality:** This captures an individual's assessment of the quality of code underlying the primary application or service they work on.
- **Individual effectiveness:** This factor captures an individual's self-assessed effectiveness and sense of accomplishment at work.
- **Valuable work:** This measures the self-assessed amount of time an individual spends doing work they feel is valuable and worthwhile.
- **Friction:** This measures the extent to which friction hinders an individual's work. Lower amounts of friction are generally considered to be a positive outcome.
- **Burnout:** This measures feelings of exhaustion and cynicism related to one's work. We consider burnout a key impediment to an individual's work.

The theory

If our findings are the structure and our analysis is the construction, then theory is the foundation. This section explains this theoretical foundation for the key pathways in our model that allow us to accurately estimate the impact of AI. To maintain clarity, this section focuses on the high-level relationships within the model. We will highlight the pathways that we anticipate require the most justification. While our analysis relies on granular relationships between specific constructs, our focus here remains on the overarching theoretical connections. Each pathway we highlight is grounded in established literature, qualitative work, and subject-matter expertise, and reinforced by a decade of our own research into what drives high performance.

Service traits → Software delivery performance

A service's inherent traits dictate the challenges of its delivery. An older service, for example, often carries significant technical debt that creates friction and slows delivery.[4,5] A service's criticality, on the other hand, acts as a powerful catalyst for organizational attention and resources. This focus may streamline processes and invest in advanced practices like automated testing and Site Reliability Engineering, simultaneously improving both speed and stability.

Non-critical services are often starved of this investment, leading to neglect, accumulating risk, and creating a drag on performance that only becomes visible when it is too late. An AI-infused service introduces many of the challenges of MLOps,[6] each fundamentally altering the stability and speed of the delivery pipeline.

Process and practice → (Multiple outcomes)

DORA's research has demonstrated, over the last decade, that the way people and teams work dictates their ability to ship software, work effectively, collaborate, and ultimately build great products. While the specific practices explored this year may be different and the development world is now inundated with AI, this fundamental principle holds.

Well-designed processes ...

1. Turn software throughput and stability into repeatable outcomes,[7]
2. Reduce coordination overhead and allow teams to spend energy on developing and learning instead of fire-fighting and stifling process,[8,9]
3. Reduce cognitive load and buffer individuals from stress,[10,11]
4. Help ideas become a reality without compromising security, software delivery performance, or reliability.[12,13]

Team performance → Individual outcomes

A well-functioning team doesn't just deliver products; it carries the individual.[14] Collaborative, reliable, and efficient teams bolster and amplify an individual's performance. A team lacking in those traits stifles and constrains an individual, creating demands that drain them.[15,16] In this way, a high-performing team provides both the conditions for good work and the pathways for that work to make an impact.

Environment traits → Individual outcomes

The work environment exerts forces on individuals that impact how they work and how they experience work. We can understand these forces through the Job Demands-Resources (JD-R) model, which separates factors that cause stress (demands) from those that enable success (resources).[17] Larger organizations, for example, often create demands like navigating bureaucracy and coping with shifting priorities.[18] Industries all have unique demands that could change the prevalence and experience of burnout.[19]

Conclusion

Further, certain industries might be facing external pressures that create stressors due to a sense of uncertainty.[20,21] Organizations also vary dramatically in terms of the availability of resources (for example, tools) and the stability of priorities. A lack of resources makes work difficult. Separately, unstable priorities create moving targets that de-incentivize ambitious, long-term projects.[22]

The theory outlined in this chapter is the foundation upon which our analysis rests.

We have focused on the model's most critical pathways rather than providing an exhaustive justification for every link. This is a deliberate choice to offer clarity and transparency into the core assumptions that guide our conclusions.

By providing this blueprint, we invite you to rigorously evaluate our findings, supplement your own thinking, and apply these causal stories to the challenges you face.

This model is our map; we encourage you to use it, question it, and help us improve it.

---

1. Han, Byung-Chul. *The Agony of Eros, Vol. 1* (MiT Press, 2017), 47.
2. This approach was adopted in 2023.
3. We tested 15 constructs total in this space.
4. Forsgren, Nicole, Jez Humble, and Gene Kim. *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations* (IT Revolution, 2018).
5. Our 2022 and 2024 reports dive into this.
6. Sculley, David, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. "Hidden technical debt in machine learning systems." Advances in neural information processing systems, 2015. 28.
7. We don't want to cite ourselves, but 10+ years of our research has underscored this point.
8. MacCormack, Alan, John Rusnak, and Carliss Y. Baldwin. "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code." Management Science, 2006, 52, no. 7. 1015–1030.
9. Hackman, J. Richard. Leading Teams: Setting the Stage for Great Performances (Harvard Business Press, 2002).
10. Bakker, Arnold B., and Evangelia Demerouti. "The job demands-resources model: State of the art." Journal of Managerial Psychology, 2007, 22, no. 3. 309–328.
11. Demerouti, Evangelia, Bakker, Arnold B., Nachreiner, Friedhelm, and Schaufeli, Wilmar B. (2001). "The job demands-resources model of burnout." Journal of Applied Psychology, 2001, 86, no. 3. 499–512.
12. Forsgren, Nicole, Jez Humble, and Gene Kim. *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations* (IT Revolution, 2018).
13. MacCormack, Alan, John Rusnak, and Carliss Y. Baldwin. "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code." Management Science, 2006, 52, no. 7. 1015–1030.
14. Hackman, J. Richard. *Leading Teams: Setting the Stage for Great Performances* (Harvard Business Press, 2002).
15. Demerouti, Evangelia, Bakker, Arnold B., Nachreiner, Friedhelm, and Schaufeli, Wilmar B. (2001). "The job demands-resources model of burnout." Journal of Applied Psychology, 2001, 86, no. (3). 499–512.
16. Bakker, Arnold B., and Evangelia Demerouti. "The job demands-resources model: State of the art." Journal of Managerial Psychology, 2007, 22, no. 3. 309–328.
17. Bakker, Arnold B., and Evangelia Demerouti. "The job demands-resources model: State of the art." Journal of Managerial Psychology, 2007, 22, no. 3. 309–328.
18. Aiken, Michael, and Jerald Hage. "Organizational alienation: A comparative analysis." American Sociological Review, 1966, 31, no. 4. 497–507. They suggest that a larger organization might create more formal rules.
19. Maslach, Christina, and Michael P. Leiter. "Understanding the burnout experience: recent research and its implications for psychiatry." World Psychiatry, 2016, 15, no. 2). 103–111.
20. When this survey was taken, many sectors in the United States were undergoing transformations, to put it euphemistically.
21. Hobfoll, Stevan E. "Conservation of resources: A new attempt at conceptualizing stress." American Psychologist, 1989, 44, no. 3. 513–524.
22. Crawford, Eean R., Jeffery A. LePine, and Bruce Louis Rich. "Linking job demands and resources to employee engagement and burnout: A theoretical and meta-analytic review." Journal of Applied Psychology, 2010, 95, no. 5. 834–848.

Next steps

Join the DORA Community to discuss, learn, and collaborate on improving the impact of technology-driven teams and organizations. https://dora.community

Explore the capabilities that enable a climate for learning, fast flow, and fast feedback. https://dora.dev/capabilities

Read the book: *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution. https://itrevolution.com/product/accelerate

Read the book: *Team Topologies: Organizing Business and Technology Teams for Fast Flow*. IT Revolution. https://itrevolution.com/product/team-topologies/

Read the book: *The Skill Code: How to Save Human Ability in an Age of Intelligent Machines*. HarperCollins. https://www.harpercollins.com/products/the-skill-code-matt-beane

Read the book: *Flow engineering: From Value Stream Mapping to Effective Action*. IT Revolution. https://itrevolution.com/product/flow-engineering

Read publications from DORA's research program, including prior DORA Reports. https://dora.dev/publications

Review frequently asked questions about the research and the reports. https://dora.dev/faq

Read and submit changes, corrections, and clarifications to this report. https://dora.dev/publications/errata

Check if this is the latest version of the 2025 DORA Report: https://dora.dev/vc/?v=2025.1

Appendix

How outcomes were evaluated

Organizational performance: This is a high-level measure of the overall success of the organization based on characteristics like profitability, market share, and customer satisfaction.

For each of the following performance indicators, how did your organization do relative to its goals over the past year?

- Increased number of customers
- Relative market share for primary products
- Your organization's overall performance
- Your organization's overall profitability
- Achievement of organizational and mission goals
- Customer satisfaction
- Operating efficiency
- Quality of products or services provided

Team performance: This factor measures the perceived effectiveness and collaborative strength of an individual's immediate team.

How would you rate your team's current performance in the following areas?

- Delivering innovative solutions
- Adapting to change
- Effectively collaborating with each other
- Being able to rely on each other
- Efficiently working together

Product performance: This factor measures the success and quality of the products or services the team is building based on characteristics like helping users accomplish important tasks, keeping information safe, and performance metrics like latency.

For the primary service or application that you work on, how would you rate your application or service's current performance across the following areas?

- Performance metrics like latency
- Doing what it is supposed to do
- Helping people accomplish what is important to them
- Usability and ease of navigation
- Keeping user information safe
- Reliability and availability for users

Software delivery throughput: This represents the speed and efficiency of the software delivery process. See the Understanding your software delivery performance chapter for more details.

- How often does your organization deploy code to production or release it to end users?
- What is your lead time for changes (that is, how long does it take to go from code committed to code successfully running in production)?
- How long does it generally take to restore service after a change to production or release to users results in degraded service (for example, leads to service impairment or service outage) and subsequently requires remediation (for example, requires a hotfix, rollback, fix forward, or patch)?

Software delivery instability: This captures the quality and reliability of the software delivery process. See the Understanding your software delivery performance chapter for more details.

- Approximately what percentage of changes to production or releases to users result in degraded service (for example, lead to service impairment or service outage) and subsequently require remediation (for example, require a hotfix, rollback, fix forward or patch), if at all?
- Approximately what percentage of deployments in the last six months were not planned but were performed to address a user-facing bug in the application?

Code quality: This captures an individual's assessment of the quality of code underlying the primary application or service they work on.

- How would you rate the quality of code underlying the primary service or application you work on?

Individual effectiveness: This factor captures an individual's self-assessed effectiveness and sense of accomplishment at work.

- In the last three months, how effectively were you able to perform your tasks and responsibilities at work?
- In the last three months, how productive did you feel in your work?
- In the last three months, how much impact do you think your work has had?
- In the last three months, how often were you able to reach a high level of focus or achieve "flow" at work?

Valuable work: This measures the self-assessed amount of time an individual spends doing work they feel is valuable and worthwhile.

- In the last three months, approximately what percentage of your time was spent doing work that felt valuable and worthwhile?

Friction: This measures the extent to which friction hinders an individual's work.

- In the last three months, to what extent did friction hinder your work?

Burnout: This measures feelings of exhaustion and cynicism related to one's work.

In the last three months, to what extent have you experienced the following?

- Felt indifferent or cynical towards your work
- Felt burned out from your work
- Felt ineffective in your work
- Your feelings about work negatively affected your life outside of work

AI adoption

AI adoption is not a key outcome but it is a central measure used throughout the report. Here's a bit more background on how we measure it.

AI adoption: This factor measures the extent to which individuals are integrating AI into their daily work and their attitudes toward it.

In the last three months, when you encountered a problem to solve or a task to complete at work, how frequently did you use AI?

In the last three months, how much have you relied on AI at work?

In the last three months, how much did you trust the quality of the output from AI-generated code as part of your development work?

Platform engineering

Definition used for platform and platform team:

Platform: A platform is a set of capabilities that is shared across multiple applications or services. A company may have multiple overlapping platforms, but we refer to these overall as "the platform."

Platform team: A platform engineering team is a group of people dedicated to building and running the platform. A dedicated platform engineering team is not required.

Platform engineering: The software and systems engineering practice used when building a platform.

List of characteristics defining a robust platform:

To what extent does your platform(s) demonstrate the following characteristics?

The platform helps me build and run reliable applications and services.

The platform's user interface (UI) is straightforward and clean.

The platform provides the tools and information I need to work independently.

The platform helps me build and run secure applications and services.

The platform behaves in a way I would expect.

The platform helps me follow required processes (such as, code reviews, security sign-offs).

The platform provides the tools and info I need to work independently.

The platform gives me clear feedback on the outcome of my tasks.

The tasks I perform on the platform are well automated.

The platform team acts on the feedback I provide.

The platform is easy to use.

The platform effectively abstracts away the complexity of underlying infrastructure.

“State of AI-assisted Software Development” by Google LLC is licensed under CC BY-NC-SA 4.0.



Get better at getting better

dora.dev

DORA

WURSTA

Take the next step

Ready to see how [Partner name] and Google Cloud can transform your business? Our team can help you explore solutions tailored to your specific needs and guide you through implementation. Let's discuss how you can leverage Google Cloud's powerful tools to improve efficiency, drive innovation, and achieve your business goals.

Contact us today for a personalized consultation and discover how we can help you unlock new possibilities with Google Cloud/Wursta..

We're here to help you every step of the way.

Sincerely,

Wursta



Learn more at wursta.com