DORA

2024

Google Cloud

Accelerate State of DevOps

Gold Sponsors

catchpoint · chronosphere · DATADOG · Deloitte. · Excella · Gearset · liatrio · Middleware · OPSERA

10

A decade with DORA

Contents

Executive summary

DORA has been investigating the capabilities, practices, and measures of high-performing technology-driven teams and organizations for over a decade. This is our tenth DORA report. We have heard from more than 39,000 professionals working at organizations of every size and across many different industries globally. Thank you for joining us along this journey and being an important part of the research!

DORA collects data through an annual, worldwide survey of professionals working in technical and adjacent roles. The survey includes questions related to ways of working and accomplishments that are relevant across an organization and to the people working in that organization.

DORA uses rigorous statistical evaluation methodology to understand the relationships between these factors and how they each contribute to the success of teams and organizations.

This year, we augmented our survey with in-depth interviews of professionals as a way to get deeper insights, triangulate, and provide additional context for our findings. See the Methodology chapter for more details.

The key accomplishments and outcomes we investigated this year are:

Reducing burnout	Burnout is a state of emotional, physical, and mental exhaustion caused by prolonged or excessive stress, often characterized by feelings of cynicism, detachment, and a lack of accomplishment.
Flow	Flow measures how much focus a person tends to achieve during development tasks.
Job satisfaction	Job satisfaction measures someone's overall feeling about their job.
Organizational performance	This measures an organization's performance in areas including profitability, market share, total customers, operating efficiency, customer satisfaction, quality of products and services, and its ability to achieve goals.
Product performance	This measures the usability, functionality, value, availability, performance (for example, latency), and security of a product.
Productivity	Productivity measures the extent to which an individual feels effective and efficient in their work, creating value and achieving tasks.
Team performance	This measures a team's ability to collaborate, innovate, work efficiently, rely on each other, and adapt.

Key findings

AI is having broad impact

AI is producing a paradigm shift in the field of software development. Early adoption is showing some promising results, tempered by caution.

AI adoption benefits:

- Flow
- Productivity
- Job satisfaction
- Code quality
- Internal documentation
- Review processes
- Team performance
- Organizational performance

However, AI adoption also brings some detrimental effects. We have observed reductions to software delivery performance, and the effect on product performance is uncertain. Additionally, individuals are reporting a decrease in the amount of time they spend doing valuable work as AI adoption increases, a curious finding that is explored more later in this report.

Teams should continue experimenting and learning more about the impact of increasing reliance on AI.

AI adoption increases as trust in AI increases

Using generative artificial intelligence (gen AI) makes developers feel more productive, and developers who trust gen AI use it more. There is room for improvement in this area: 39.2% of respondents reported having little or no trust in AI.

User-centricity drives performance

Organizations that prioritize the end user experience produce higher quality products, with developers who are more productive, satisfied, and less likely to experience burnout.

Transformational leadership matters

Transformational leadership improves employee productivity, job satisfaction, team performance, product performance, and organizational performance while also helping decrease employee burnout.

Stable priorities boost productivity and well-being

Unstable organizational priorities lead to meaningful decreases in productivity and substantial increases in burnout, even when organizations have strong leaders, good internal documents, and a user-centric approach to software development.

Platform engineering can boost productivity

Platform engineering has a positive impact on productivity and organizational performance, but there are some cautionary signals for software delivery performance.

Cloud enables infrastructure flexibility

Flexible infrastructure can increase organizational performance. However, moving to the cloud without adopting the flexibility that cloud has to offer may be more harmful than remaining in the data center. Transforming approaches, processes, and technologies is required for a successful migration.

High-levels of software delivery performance are achievable

The highest performing teams excel across all four software delivery metrics (change lead time, deployment frequency, change fail percentage, and failed deployment recovery time) while the lowest performers perform poorly across all four. We see teams from every industry vertical in each of the performance clusters.

Applying insights from DORA

Driving team and organizational improvements with DORA requires that you assess how you're doing today, identify areas to invest in and make improvements, and have feedback loops to tell you how you're progressing. Teams that adopt a mindset and practice of continuous improvement are likely to see the most benefits. Invest in building the organizational muscles required to repeat this over time.

Findings from our research can help inform your own experiments and hypotheses. It's important to experiment and measure the impact of your changes to see what works best for your team and organization. Doing so will help you validate our findings. Expect your results to differ and please share your progress so that we all may learn from your experience.

We recommend taking an experimental approach to improvement.

1. Identify an area or outcome you would like to improve
2. Measure your baseline or current state
3. Develop a set of hypotheses about what might get you closer to your desired state
4. Agree and commit to a plan for improvement
5. Do the work
6. Measure the progress you've made
7. Repeat the process. Improvement work is achieved iteratively and incrementally

DORA COMMUNITY

You cannot improve alone!

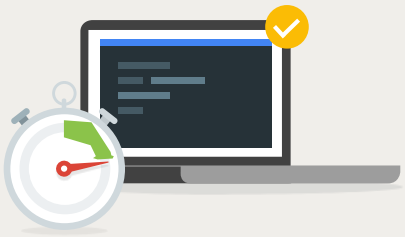
We can learn from each other's experience; an excellent forum for sharing and learning about improvement initiatives is the DORA Community **https://dora.community**.

Software delivery performance

Technology-driven teams need ways to measure performance so that they can assess how they're doing today, prioritize improvements, and validate their progress. DORA has repeatedly validated four software delivery metrics — the four keys — that provide an effective way of measuring the outcomes of the software delivery process.

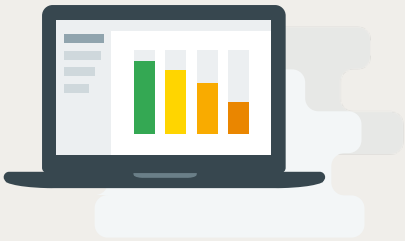
The four keys

DORA's four keys have been used to measure the throughput and stability of software changes. This includes changes of any kind, including changes to configuration and changes to code.

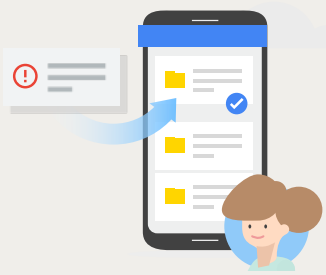


Change lead time:
the time it takes for a code commit or change to be successfully deployed to production.

Deployment frequency:
how often application changes are deployed to production.



Change fail rate:
the percentage of deployments that cause failures in production,[1] requiring hotfixes or rollbacks.



Failed deployment recovery time:
the time it takes to recover from a failed deployment.

We've observed that these metrics typically move together, the best performers do well on all four while the lowest performers do poorly.

Evolving the measures of software delivery performance

The analysis of the four key metrics has long had an outlier: change failure rate.[2] Change failure rate is strongly correlated with the other three metrics but statistical tests and methods prevent us from combining all four into one factor. A change to the way respondents answered the change failure rate question improved the connection but we felt there might be something else happening.

We have a longstanding hypothesis that the change failure rate metric works as a proxy for the amount of rework a team is asked to do. When a delivery fails, this requires the team to fix the change, likely by introducing another change.

To test this theory, we added another question this year about the rework rate for an application: "For the primary application or service you work on, approximately how many deployments in the last six months were not planned but were performed to address a user-facing bug in the application?"

Our data analysis confirmed our hypothesis that rework rate and change failure rate are related. Together, these two metrics create a reliable factor of software delivery stability.

This appears in the analysis of software performance levels, too. More than half of the teams in our study this year show differences in software throughput and software stability. These differences have led us to consider software delivery performance through two different factors:

Concept	
Software delivery performance	
Factor	
Software delivery throughput	**Software delivery stability**
Metrics used	
• **Change lead time** • **Deployment frequency** • **Failed deployment recovery time**	• **Change failure rate** • **Rework rate**

Our analysis throughout this report utilizes the software delivery performance concept and both factors at various times. All five metrics are considered for describing software delivery performance.

Change lead time, deployment frequency, and failed deployment recovery time are used when we describe software delivery throughput. This factor measures the speed of making updates of any kind, normal changes and changes in response to a failure.

Change failure rate and rework rate are used when we describe software delivery stability. This factor measures the likelihood deployments unintentionally lead to immediate, additional work.

Performance levels

Each year we ask survey respondents about the software delivery performance of the primary application or service they work on. We analyze their answers using cluster analysis, which is a statistical method that identifies responses that are similar to one another but distinct from other groups of responses.

We performed the cluster analysis on the original four software delivery metrics to remain consistent with previous years' cluster analyses.

In our analysis of software delivery performance, four clusters of responses emerged. We do not set these levels in advance, rather we let them emerge from the survey responses. This gives us a way to see a snapshot of software delivery performance across all respondents each year.

Four distinct clusters emerged from the data this year, as shown below.

Performance level	Change lead time	Deployment frequency	Change fail rate	Failed deployment recovery time	Percentage of respondents*
Elite	**Less than one day**	**On demand (multiple deploys per day)**	**5%**	**Less than one hour**	**19% (18-20%)**
High	**Between one day and one week**	**Between once per day and once per week**	**20%**	**Less than one day**	**22% (21-23%)**
Medium	**Between one week and one month**	**Between once per week and once per month**	**10%**	**Less than one day**	**35% (33-36%)**
Low	**Between one month and six months**	**Between once per month and once every six months**	**40%**	**Between one week and one month**	**25% (23-26%)**

*89% uncertainty interval

Throughput or stability?

Within all four clusters, throughput and stability are correlated. This correlation persists even in the medium performance cluster (orange), where throughput is lower and stability is higher than in the high performance cluster (yellow). This suggests that factors besides throughput and stability influence cluster performance. The medium performance cluster, for example, may benefit from shipping changes more frequently.

Which is better, more frequent deployments or fewer failures when deploying?

There may not be a universal answer to this question. It depends on the application or service being considered, the goals of the team working on that application, and most importantly the expectations of the application's users.

We made a decision to call the faster teams "high performers," and the slower but more stable teams "medium performers." This decision highlights one of the potential pitfalls of using these performance levels: Improving should be more important to a team than reaching a particular performance level. The best teams are those that achieve elite ***improvement***, not necessarily elite ***performance***.

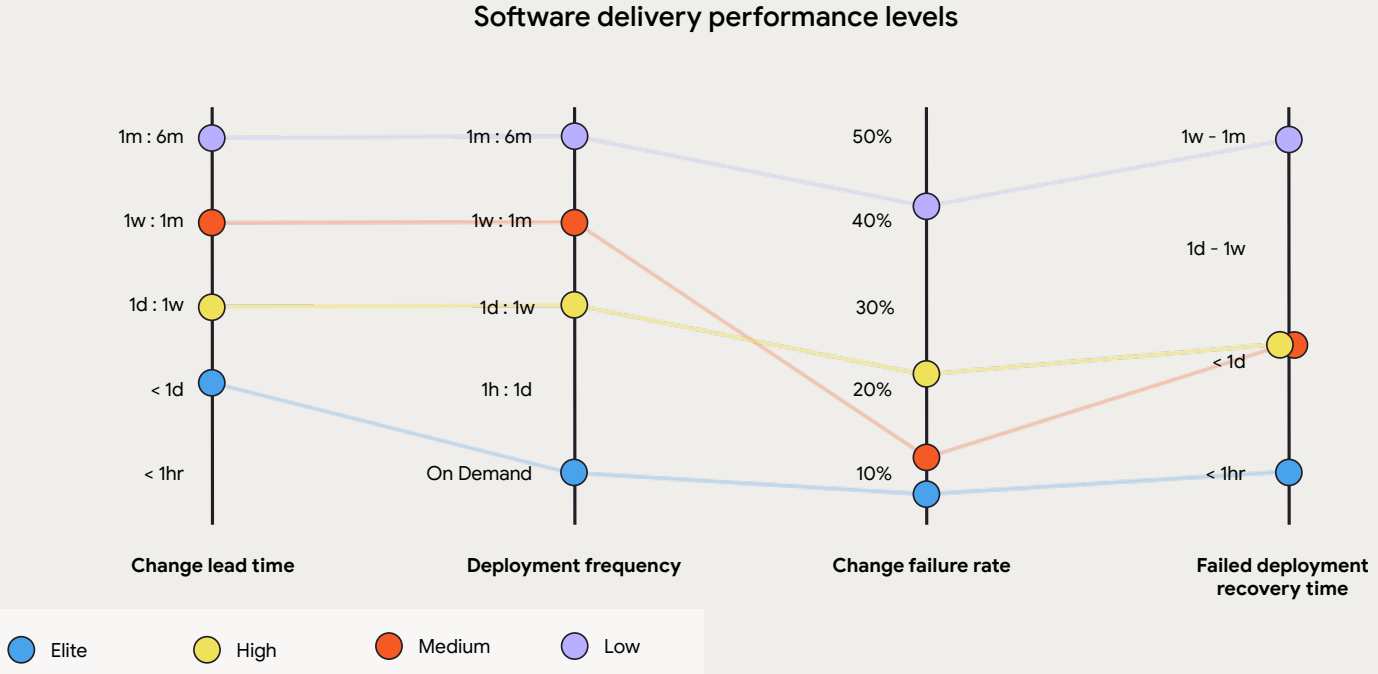


Figure 1: Software delivery performance levels

When compared to low performers, elite performers realize

127x	182x	8x	2293x
faster lead time	more deployments per year	lower change failure rate	faster failed deployment recovery times

How to use the performance clusters

The performance clusters provide benchmark data that show the software delivery performance of this year's survey respondents. The clusters are intended to help inspire all that elite performance is achievable.

More important than reaching a particular performance level, we believe that teams should focus on improving performance overall. The best teams are those that achieve elite ***improvement***, not necessarily elite ***performance***.

Industry does not meaningfully affect performance levels

Our research rarely[3] finds that industry is a predictor of software delivery performance; we see high-performing teams in every industry vertical. This isn't to suggest that there are no unique challenges across industries, but no one industry appears to be uniquely encumbered or uniquely capable when it comes to software delivery performance.

Using the software delivery performance metrics

Each application or service has its own unique context. This complexity makes it difficult to predict how any one change may affect the overall performance of the system. Beyond that, it is nearly impossible to change only one thing at a time in an organization. With this complexity in mind, how can we use the software delivery performance metrics to help guide our improvement efforts?

Start by identifying the primary application or service you would like to measure and improve. We then recommend gathering the cross-functional team responsible for this application to measure and agree on its current software delivery performance. The DORA Quick Check (https://dora.dev/quickcheck) can help guide a conversation and set this baseline measurement. Your team will need to understand what is preventing better performance.

One effective way to find these impediments is to complete a value stream mapping exercise[4] with the team.

Next, identify and agree on a plan for improvement. This plan may focus on improving one of the many capabilities that DORA has researched[5] or may be something else that is unique to your application or organization.

With this plan in-hand, it's now time to do the work! Dedicate capacity to this improvement work and pay attention to the lessons learned along the way.

After the change has had a chance to be implemented and take hold, it's now time to re-evaluate the four keys. How have they changed after the team implemented the change? What lessons have you learned?

Repeating this process will help the team build a practice of continuous improvement.

Remember: change does not happen overnight. An iterative approach that enables a climate for learning, fast flow, and fast feedback[6] is required.

---

1. We consider a deployment to be a change failure only if it causes an issue after landing in production, where it can be experienced by end users. In contrast, a change that is stopped on its way to production is a successful demonstration of the deployment process's ability to detect errors.
2. Forsgren, Nicole, Jez Humble, and Gene Kim. 2018. Accelerate: The Science Behind DevOps : Building and Scaling High Performing Technology Organizations. IT Revolution Press. pp. 37-38
3. The 2019 Accelerate State of DevOps (p 32) report found that the retail industry saw significantly better software delivery performance. https://dora.dev/research/2019/dora-report/2019-dora-accelerate-state-of-devops-report.pdf#page=32
4. https://dora.dev/guides/value-stream-management/
5. https://dora.dev/capabilities
6. https://dora.dev/research

Artificial intelligence: Adoption and attitudes

Takeaways

The vast majority of organizations across all industries surveyed are shifting their priorities to more deeply incorporate AI into their applications and services. A corresponding majority of development professionals are relying on AI to help them perform their core role responsibilities — and reporting increases in productivity as a result. Development professionals' perceptions that using AI is necessary for remaining competitive in today's market is pervasive and appears to be an important driver of AI adoption for both organizations and individual development professionals.

Introduction

It would be difficult to ignore the significant impact that AI has had on the landscape of development work this year, given the proliferation of popular news articles outlining its effects, from good[1] to bad[2] to ugly.[3] So, while AI was only discussed as one of many technical capabilities affecting performance in our 2023 Accelerate State of DevOps Report,[4] this year we explore this topic more fully.

As the use of AI in professional development work moves rapidly from the fringes to ubiquity, we believe our 2024 Accelerate State of DevOps Report represents an important opportunity to assess the adoption, use, and attitudes of development professionals at a critical inflection point for the industry.

Findings

Adopting Artificial Intelligence

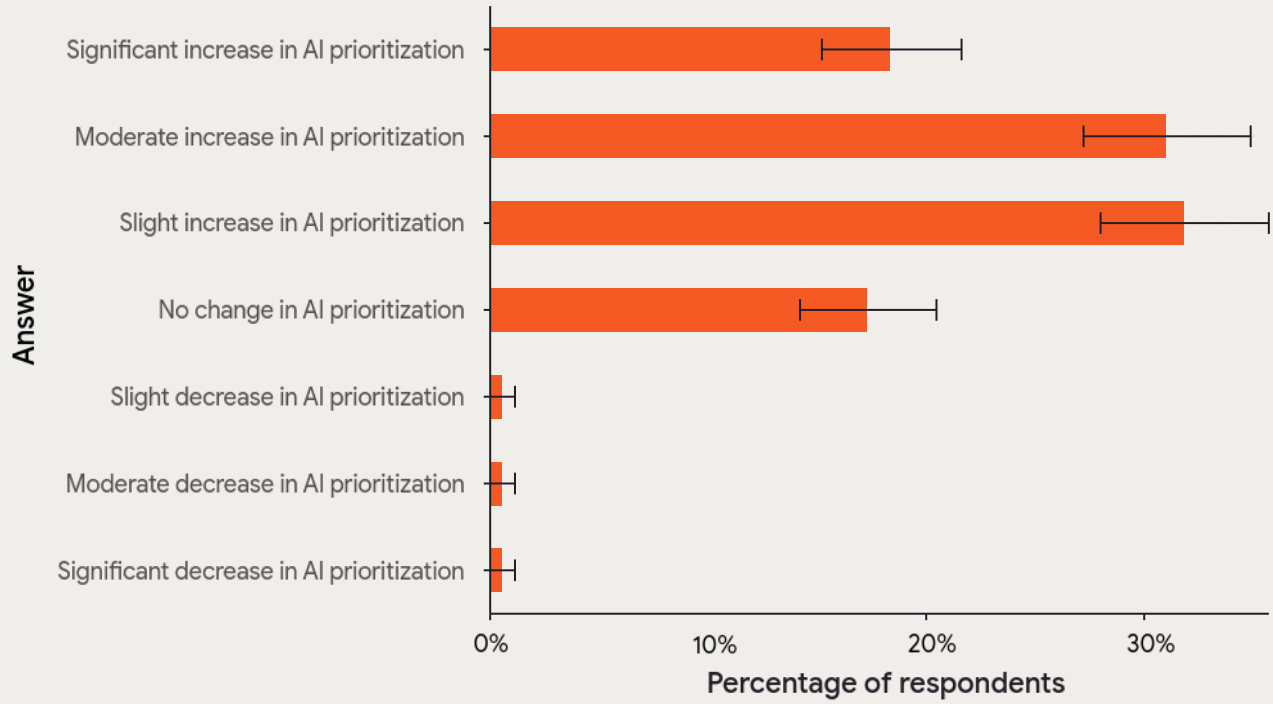
Findings on the adoption of AI suggest a growing awareness that AI is no longer "on the horizon," but has fully arrived and is, quite likely, here to stay.

Organizational adoption of Artificial Intelligence

The vast majority of respondents (81%) reported that their organizations have shifted their priorities to increase their incorporation of AI into their applications and services. 49.2% of respondents even described the magnitude of this shift as being either "moderate" or "significant."

Notably, 3% of respondents reported that their organizations are decreasing focus on AI — within the margin of error of our survey. 78% of respondents reported that they trusted their organizations to be transparent about how they plan on using AI as a result of these priority shifts. This data is visualized in Figure 2.

Changes in organizational priorities concerning AI



Error bar represents 89% uncertainty interval

Figure 2: Respondents' perceptions of their organizations' shifts in priorities toward or away from incorporation of AI into their applications and services.

Participants from all surveyed industries reported statistically identical levels of reliance on AI in their daily work, which suggests that this rapid adoption of AI is unfolding uniformly across all industry sectors. This was somewhat surprising to us. Individual industries can vary widely with respect to their levels of regulatory constraints and historical pace of innovation, each of which can impact rates of technology adoption.

However, we did find that respondents working in larger organizations report less reliance on AI in their daily work than respondents working in smaller organizations, which is consistent with prior literature indicating larger firms more slowly adapt to technological change because of their higher organizational complexities and coordination costs.[5]

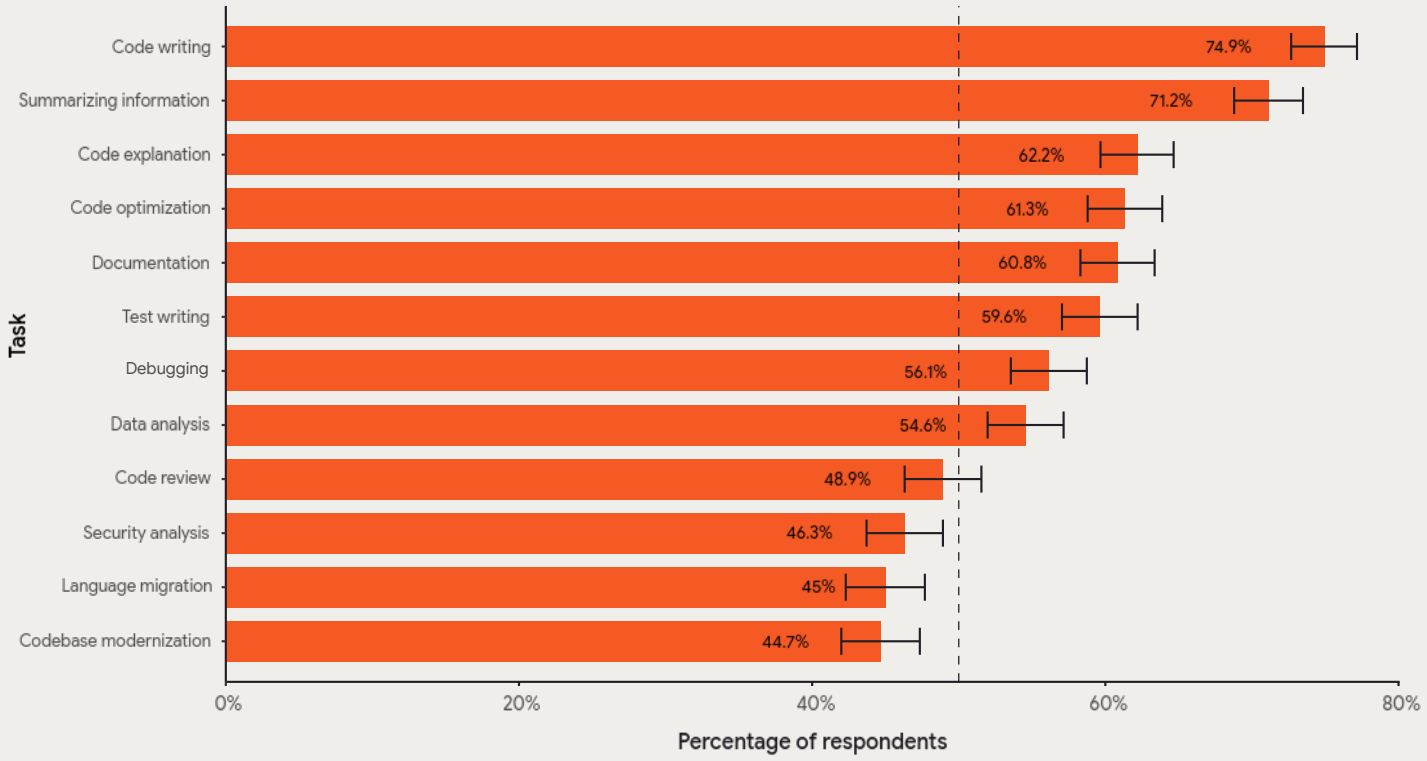
Individual adoption of artificial intelligence

At the individual level, we found that 75.9% of respondents are relying, at least in part, on AI in one or more of their daily professional responsibilities. Among those whose job responsibilities include the following tasks, a majority of respondents relied on AI for:

1. Writing code
2. Summarizing information
3. Explaining unfamiliar code
4. Optimizing code
5. Documenting code
6. Writing tests
7. Debugging code
8. Data analysis

Of all tasks included in our survey responses, the most common use cases for AI in software development work were writing code and summarizing information, with 74.9% and 71.2% of respondents whose job responsibilities include these tasks relying on AI to perform them — at least in part. This data is visualized in Figure 3.

Task reliance on AI



Error bar represents 89% credibility interval

Figure 3: Percentage of respondents relying on AI, at least in part, to perform twelve common development tasks

Chatbots were the most common interface through which respondents interacted with AI in their daily work (78.2%), followed by external web interfaces (73.9%), and AI tools embedded within their IDEs (72.9%). Respondents were less likely to use AI through internal web interfaces (58.1%) and as part of automated CI/CD pipelines (50.2%).

However, we acknowledge that respondents' awareness of AI used in their CI/CD pipelines and internal platforms likely depends on the frequency with which they interface with those technologies. So, these numbers might be artificially low.

We found that data scientists and machine learning specialists were more likely than respondents holding all other job roles to rely on AI. Conversely, hardware engineers were less likely than respondents holding all other job roles to rely on AI, which might be explained by the responsibilities of hardware engineers differing from the above tasks for which AI is commonly used.

Drivers of adoption of artificial intelligence

Interview participants frequently linked the decision to adopt AI to competitive pressures and a need to keep up with industry standards for both organizations and developers, which are increasingly recognized to include proficiency with AI.

For several participants' organizations, using AI at all was seen as "a big marketing point" (P3)[6] that could help differentiate their firm from competitors. Awareness that competitors are beginning to adopt AI in their own processes even prompted one firm to forgo the typical "huge bureaucracy" involved in adopting new technology because they felt an urgency to adopt AI, questioning "what if our competitor takes those actions before us?" (P11).

At the individual level, many participants linked their adoption of AI to the sentiment that proficiency with using AI in software development is "kind of, like, the new bar for entry as an engineer" (P9). Several participants suggested fellow developers should rapidly adopt AI in their development workflow, because "there's so much happening in this space, you can barely keep up... I think, if you don't use it, you will be left behind quite soon" (P4).

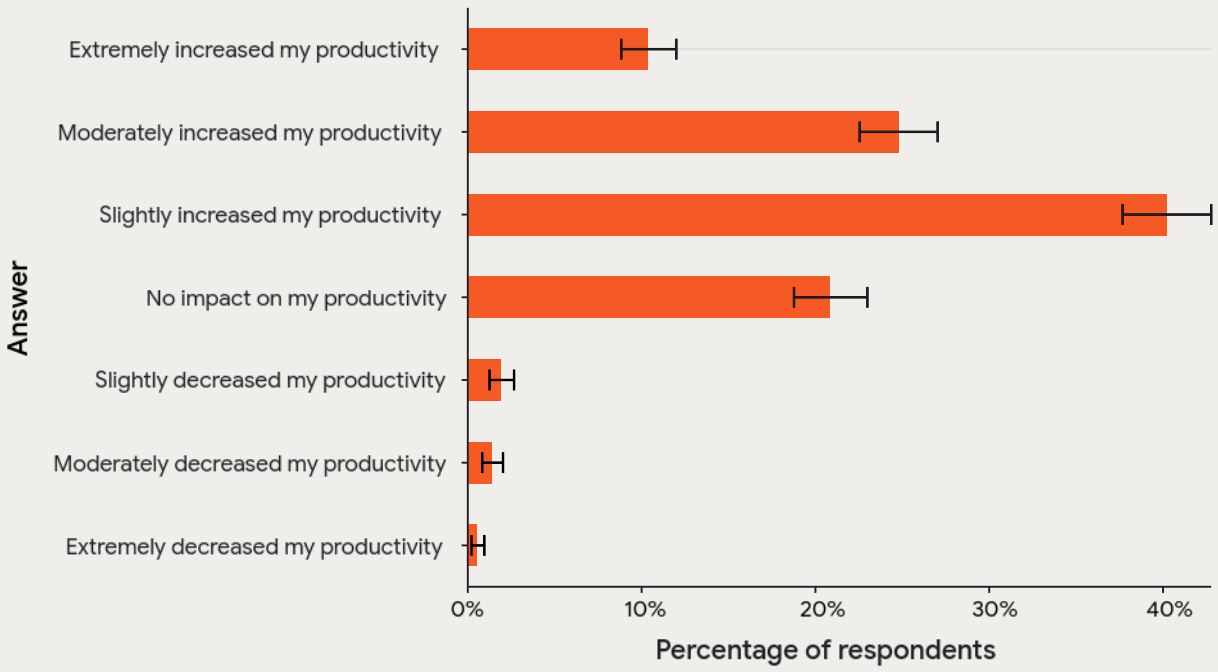
Perceptions of artificial intelligence

Performance improvements from artificial intelligence

For the large number of organizations and developers who are adopting it, the benefits of using AI in development work appear to be quite high. Seventy-five percent of respondents reported positive productivity gains from AI in the three months preceding our survey, which was fielded in early 2024.

Notably, more than one-third of respondents described their observed productivity increases as either moderate (25%) or extreme (10%) in magnitude. Fewer than 10% of respondents reported negative impacts of even a slight degree on their productivity because of AI. This data is visualized in Figure 4.

Perceptions of productivity changes due to AI



Error bar represents 89% uncertainty interval

Figure 4: Respondents' perceptions of AI's impacts on their productivity.

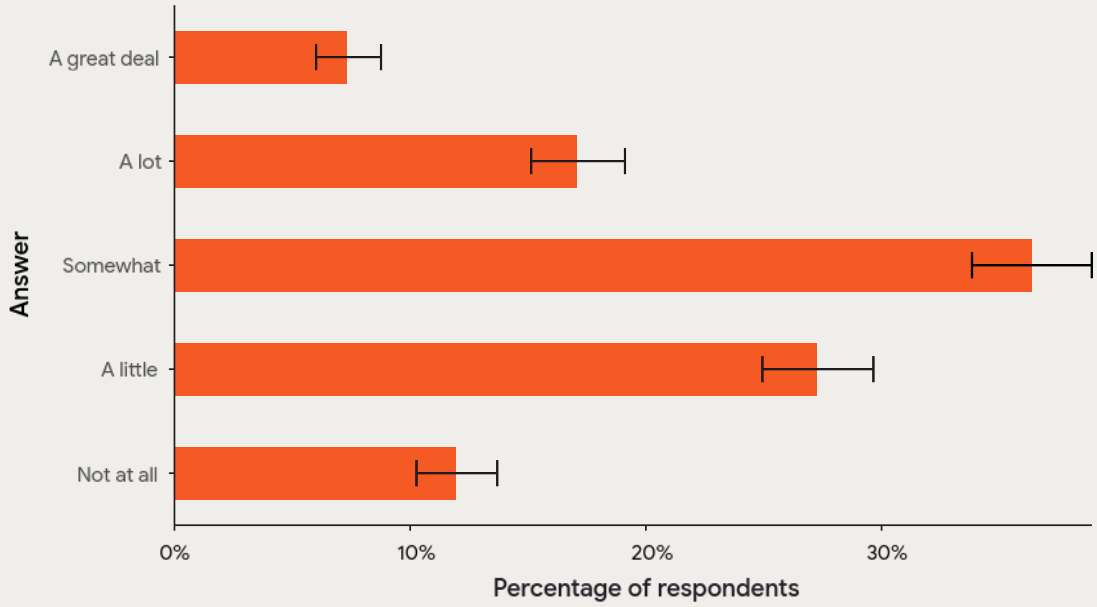
Across roles, respondents who reported the largest productivity improvements from AI were security professionals, system administrators, and full-stack developers. Although they also reported positive productivity improvement, mobile developers, site reliability engineers, and project managers reported lower magnitudes of productivity benefits than all other named roles.

Although we suspected that the novelty of AI in development work, and corresponding learning curve, might inhibit developers' ability to write code, our findings did not support that hypothesis. Only 5% of respondents reported that AI had inhibited their ability to write code to any degree. In fact, 67% of respondents reported at least some improvement to their ability to write code as a result of AI-assisted coding tools, and about 10% have observed "extreme" improvements to their ability to write code because of AI.

Trust in AI-generated code

Participants' perceptions of the trustworthiness of AI-generated code used in development work were complex. While the vast majority of respondents (87.9%) reported some level of trust in the quality of AI-generated code, the degree to which respondents reported trusting the quality of AI-generated code was generally low, with 39.2% reporting little (27.3%) or no trust (11.9%) at all. This data is visualized in Figure 5.

Trust in quality of AI-generated code



Error bar represents 89% uncertainty interval

Figure 5: Respondents' reported trust in the quality of AI-generated code.

Given the evidence from the survey that developers are rapidly adopting AI, relying on it, and perceiving it as a positive performance contributor, we found the overall lack of trust in AI surprising. It's worth noting that during our interviews, many of our participants indicated that they were willing to, or expected to, tweak the outputs of the AI-generated code they used in their professional work.

One participant even likened the need to evaluate and modify the outputs of AI-generated code to "the early days of StackOverflow, [when] you always thought people on StackOverflow are really experienced, you know, that they will know exactly what to do. And then, you just copy and paste the stuff, and things explode" (P2).

Perhaps because this is not a new problem, participants like P3 felt that their companies are not "worried about, like, someone just copy-and-pasting code from Copilot or ChatGPT [because of] having so many layers to check it" with their existing code-quality assurance processes.

We hypothesize that developers do not necessarily expect absolute trust in the accuracy of AI-generated code, nor does absolute trust appear to be required for developers to find AI-generated code useful. Rather, it seems that mostly-correct AI-generated code that can be perfected with some tweaks is acceptable, sufficiently valuable to motivate widespread adoption and use, and compatible with existing quality assurance processes.

Expectations for AI's future

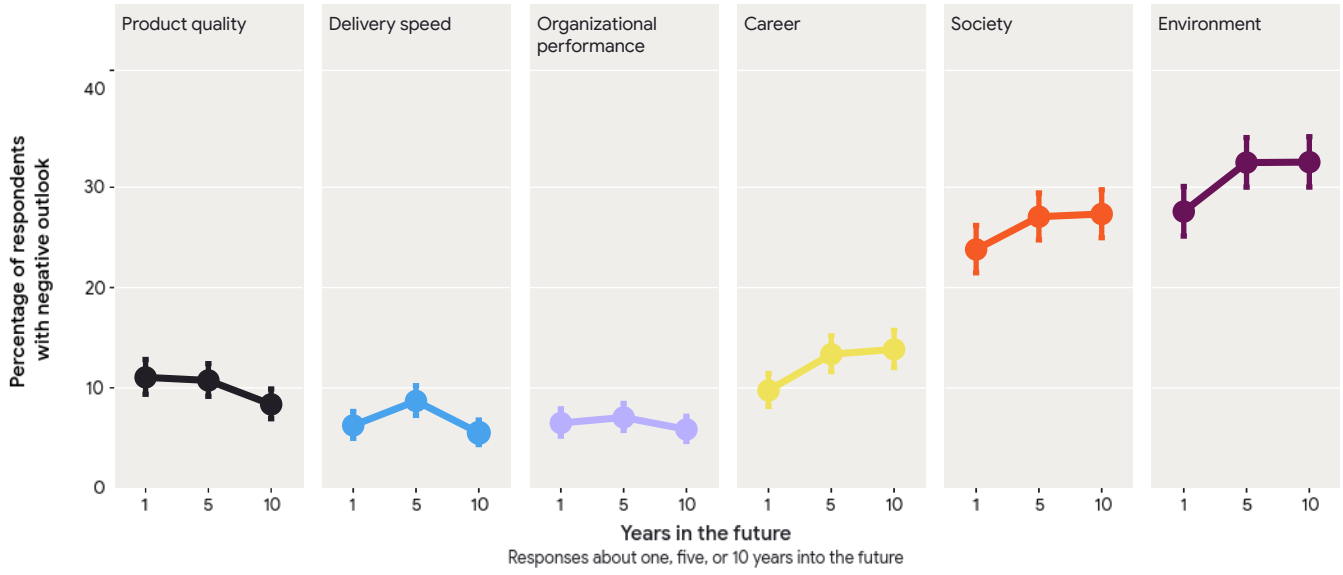
Overall, our findings indicate AI has already had a massive impact on development professionals' work, a trend we expect to continue to grow. While it would be impossible to predict exactly how AI will impact development — and our world — in the future, we asked respondents to speculate and share their expectations about the impacts of AI in the next one, five, and 10 years.

Respondents reported quite positive impacts of AI on their development work in reflecting on their recent experiences, but their predictions for AI's future impacts were not as hopeful.

Optimistically, and consistent with our findings that AI has positively impacted development professionals' performance, respondents reported that they expect the quality of their products to continue to improve as a result of AI over the next one, five, and 10 years.

However, respondents also reported expectations that AI will have net-negative impacts on their careers, the environment, and society, as a whole, and that these negative impacts will be fully realized in about five years time. This data is visualized in Figure 6.

Expected negative impacts of AI



Error bar represents 89% credibility interval

Figure 6: Respondents' expectations about AI's future negative impacts in the next one, five, and 10 years.

Interview participants held similarly mixed feelings about the future impacts of AI as our survey respondents. Some wondered about future legal actions in a yet-to-be-decided regulatory landscape, worrying they might "be on the wrong side of it, if things get decided" (P3).

Others echoed long-held anxieties and asked, "Is it going to replace people? Who knows? Maybe." (P2), while their peers dismissed their fears by drawing parallels to the past, when "people used to say 'Oh, Y2K! Everything will be doomed!' Blah, blah... because it was a new thing, at that time. [But,] nothing got replaced. In fact, there were more jobs created. I believe the same thing will happen with AI" (P1).

The future effects AI will have on our world remain unclear. But, this year, our survey strongly indicates that AI has produced an unignorable paradigm shift in the field of software development. So far, the changes have been well-received by development professionals.

1. https://www.sciencedaily.com/releases/2024/03/240306144729.htm
2. https://tech.co/news/list-ai-failures-mistakes-errors
3. https://klyker.com/absurd-yoga-poses-generated-by-ai/
4. https://dora.dev/dora-report-2023
5. Rogers, Everett M., Arvind Singhal, and Margaret M. Quinlan. "Diffusion of innovations." An integrated approach to communication theory and research. Routledge, 2014. 432-44, Tornatzky, L. G., & Fleischer, M. (1990). The processes of technological innovation. Lexington, MA: Lexington Books
6. (P[N]), for example (P1), indicates pseudonym of interview participants.

Exploring the downstream impact of AI

Takeaways

This chapter investigates the impact of AI adoption across the spectrum, from individual developers to entire organizations. The findings reveal a complex picture with both clear benefits and unexpected drawbacks. While AI adoption boosts individual productivity, flow, and job satisfaction, it may also decrease time spent on valuable work.

Similarly, AI positively impacts code quality, documentation, and review processes, but surprisingly, these gains do not translate to improved software delivery performance. In fact, AI adoption appears detrimental in this area, while its effect on product performance remains negligible.

Despite these challenges, AI adoption is linked to improved team and organizational performance. This chapter concludes with a call to critically evaluate AI's role in software development and proactively adapt its application to maximize benefits and mitigate unforeseen consequences.

The AI moment & DORA

Estimates suggest that leading tech giants will invest approximately $1 trillion on the development of AI in the next five years.[1] This aligns well with a statistic presented in the "Artificial intelligence: Adoption and attitudes" chapter that 81% of respondents say their company has shifted resources into developing AI.

The environmental impacts of AI further compound the costs. Some estimates suggest that by 2030, AI will drive an increase in data center power demand by 160%.[2] The training of an AI model can add up to roughly "the yearly electricity consumption of over 1,000 U.S. households".[3] It is no surprise that more than 30% of respondents think AI is going to be detrimental to the environment.

Beyond the development and environmental costs, we have the potential for adoption costs.

This could come in many forms, from productivity decreases to the hiring of specialists. These adoption costs could also come at a societal level. Over a third of respondents believe AI will harm society in the coming decade. Given these costs, it seems natural for people to have a deep curiosity about the returns.

This curiosity has manifested itself in a wealth of media, articles, and research whose sentiment and data are both mixed, at least to some extent.

Some believe that AI has dramatically enhanced the ability of humanity,[4] others suggest that AI is little more than a benign tool for helping with homework,[5] and some fear that AI will be the downfall of humanity.[6]

Evidence for proximal outcomes, such as the ability to successfully complete a particular task, is largely positive.[7] When the outcome becomes more distant, such as a team's codebase, the results start becoming a little less clear and a little less positive. For example, some research has suggested that code churn may double from the pre-2021 baseline.[8]

The challenge of understanding these downstream effects is unsurprising. The further away the effect is from the cause, the less pronounced and clear the connection.

Evaluating the downstream effects of AI mimics quantifying the effect of a rock thrown into a lake. You can most easily attribute the ripples closest to the impact point of the rock in the water, but the farther from the entry point you go, the less pronounced the effect of the rock is and the harder it is to ascribe waves to its impact.

AI is essentially a rock thrown into a stormy sea of other processes and dynamics. Understanding the extent of the waves caused by AI (or any technology or practice) is a challenge. This may be part of the reason the industry has struggled to adopt a principled set of measurement and analytic frameworks for understanding the impact of AI.[9]

Our approach is specifically designed to be useful for these types of challenges. DORA is designed to understand the utility or disutility of a practice. We've explored the downstream impacts of myriad practices over the last 10 years, including security practices, transformational leadership, generative cultures, documentation practices, continuous integration, continuous delivery, and user-centricity.[10]

We believe that DORA's approach[11] can help us learn about AI's impact, especially as we explore the effects of AI across many outcomes.

Measuring AI adoption

The first challenge of capturing the impact of adopting AI is measuring the adoption of AI. We determined measuring usage frequency is likely not as meaningful as measuring reliance for understanding AI's centrality to development workflows. You might only do code reviews or write documentation a few times a month or every couple of months, but you see these tasks as critically important to your work.

Conversely, just because you use AI frequently does not mean that you are using AI for work that you consider important or central to your role.

Given this, we asked respondents about their reliance on AI in general and for particular tasks. The previous chapter details the survey results and their interpretation.

Using factor analysis, we found our "general" AI reliance survey item had high overlap with reported AI reliance on the following tasks:

- Code Writing
- Summarizing information
- Code explanation
- Code optimization
- Documentation
- Test writing

The strong commonality and covariance among these seven items suggests an underlying factor that we call AI adoption.

AI's impact on individuals is a story of clear benefits (and some potential tradeoffs)

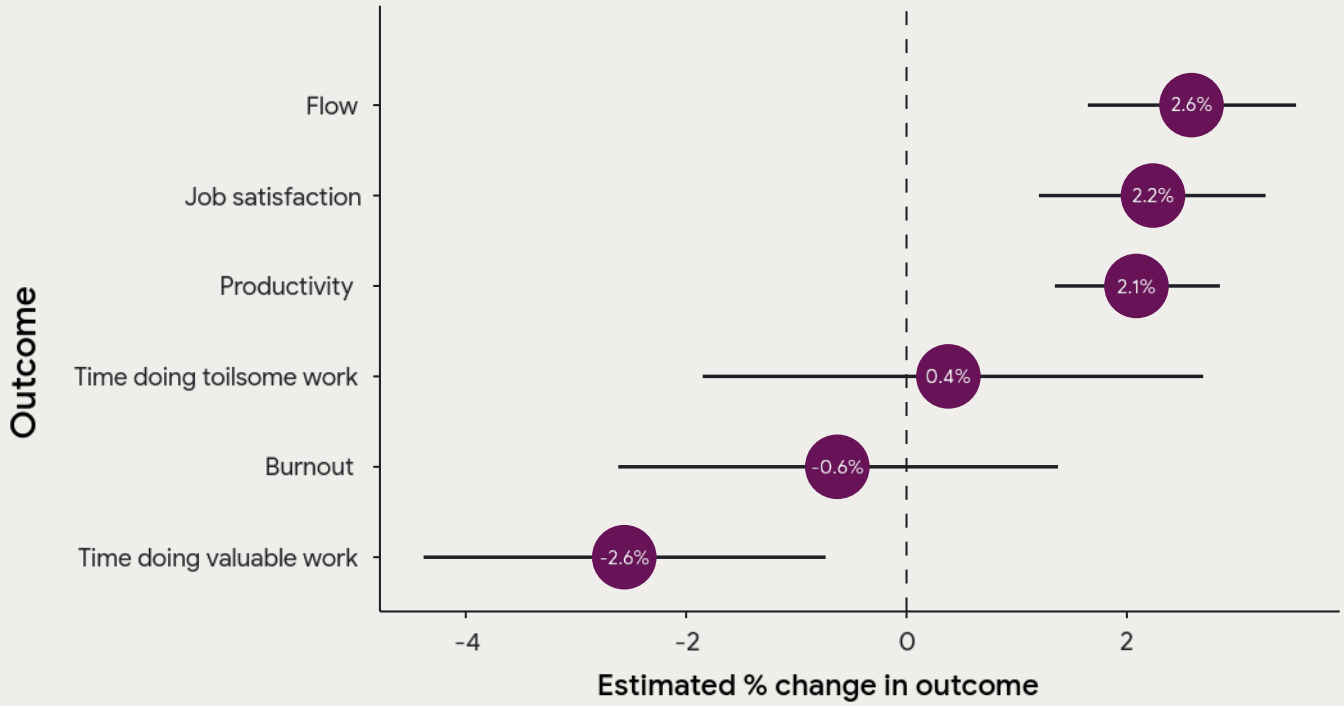
As we do every year, we measured a variety of constructs related to an individual's success and well-being:

Job satisfaction	A single item designed to capture someone's overall feeling about their job.
Burnout	A factor that encapsulates the multifaceted nature of burnout, encompassing its physical, emotional, and psychological dimensions, as well as its impact on personal life.
Flow	A single item designed to capture how much focus a person tends to achieve during development tasks.
Productivity	A factor score designed to measure the extent an individual feels effective and efficient in their work, creating value and achieving tasks.
Time doing toilsome work	A single item measuring the percentage of an individual's time spent on repetitive, manual tasks that offer little long-term value.
Time doing valuable work	A single item measuring the percentage of an individual's time spent on tasks that they consider valuable.

We wanted to figure out if the way respondents answered these questions changes as a function of adopting AI. The results suggest that is often the case.

Figure 7 is a visualization that shows our best estimates about the impact of adopting AI on an individual's success and well-being.

If an individual increases AI adoption by 25%...



Point = estimated value
Error bar = 89% uncertainty interval
Figure 7: Impacts of AI adoption on individual success and well-being

The clear benefits

The story about the benefit of adopting AI for individuals is largely favorable, but like any good story, has some wrinkles. **What seems clear is that AI has a substantial and beneficial impact on flow, productivity, and job satisfaction (see Figure 7)**.

Productivity, for example, is likely to increase by approximately 2.1% when an individual's AI adoption is increased by 25% (see Figure 7). This might seem small, but this is at the individual-level. Imagine this pattern extended across tens of developers, or even tens of thousands of developers.

This pattern is what we expected. We believe it emerged in part thanks to AI's ability to synthesize disparate sources of information and give a highly personalized response in a single location. Doing this on your own takes time, lots of context switching, and is less likely to foster flow.

Given the strong connection that productivity and flow have with job satisfaction, it shouldn't be surprising that we see AI adoption leads to higher job satisfaction.

The potential tradeoffs

Here is where the story gets a little complicated. One value proposition for adopting AI is that it will help people spend more time doing valuable work. That is, by automating the manual, repetitive, toilsome tasks, we expect respondents will be free to use their time on "something better." However, our data suggest that increased AI adoption may have the opposite effect—reducing reported time spent doing valuable work—while time spent on toilsome work appears to be unaffected.

Markers of respondents' well-being, like flow, job satisfaction, and productivity have historically been associated with time spent doing valuable work. So, observed increases in these measures independently of decreases in time spent on valuable work are surprising.

A good explanation of these patterns will need to wrestle with this seeming incongruity. A good explanation of a movie cannot ignore a scene that contradicts the explanation. A good explanation of a book cannot ignore a chapter that doesn't fit neatly into the explanation. Similarly, a good explanation of these patterns cannot just focus on a subset of the patterns that allows us to tell a simple story.

There are innumerable hypotheses that could fit the data, but we came up with a hypothesis that seems parsimonious with flow, productivity, and job satisfaction benefitting from AI while time spent doing valuable work decreases and toil remains unchanged.

We call our hypothesis the vacuum hypothesis. By increasing productivity and flow, AI is helping people work more efficiently. This efficiency is helping people finish up work they consider valuable faster.

This is where the vacuum is created; there is extra time. AI does not steal value from respondents' work, it expedites its realization.

Wait, what is valuable work?

To make sense of these counterintuitive findings we explored more deeply what types of work respondents judge to be valuable or toilsome.

Traditional wisdom, our past reports, and qualitative data from our interviews suggest that respondents find development-related tasks, like coding, to be valuable work, while less-valuable, even toilsome, work typically includes tasks associated with organizational coordination, like attending meetings. Within this categorization scheme, AI is better poised to assist with "valuable" work than "toilsome" work, as defined by respondents.

We turned to qualitative data from our interviews and found that, when responding to the moderator's question of whether or not they would consider their work "meaningful," participants frequently measured the value of their work in relation to the impact of their work on others.

This is solidified by two years of past DORA evidence of the extremely beneficial impact of user-centricity on job satisfaction.

For example, when describing a recent role shift, P10[12] indicated making the decision because "It helps me impact more people. It helps me impact more things." Similarly, P11 noted "if you build something from scratch and see it's delivered to a consumer or customer, you can feel that achievement, you can say to yourself, 'Yeah! I delivered this and people use that!'"

Understanding that the "meaningfulness" of development work is derived from the impact of the solution created—not directly from the writing of the code—helps explain why we observed respondents spending less time on valuable work, while also feeling more satisfied with their jobs.

While AI is making the tasks people consider valuable easier and faster, it isn't really helping with the tasks people don't enjoy. That this is happening while toil and burnout remain unchanged, obstinate in the face of AI adoption, highlights that AI hasn't cracked the code of helping us avoid the drudgery of meetings, bureaucracy, and many other toilsome tasks (Figure 8).

The good news is that AI hasn't made it worse, nor has it negatively affected respondents' well-being.

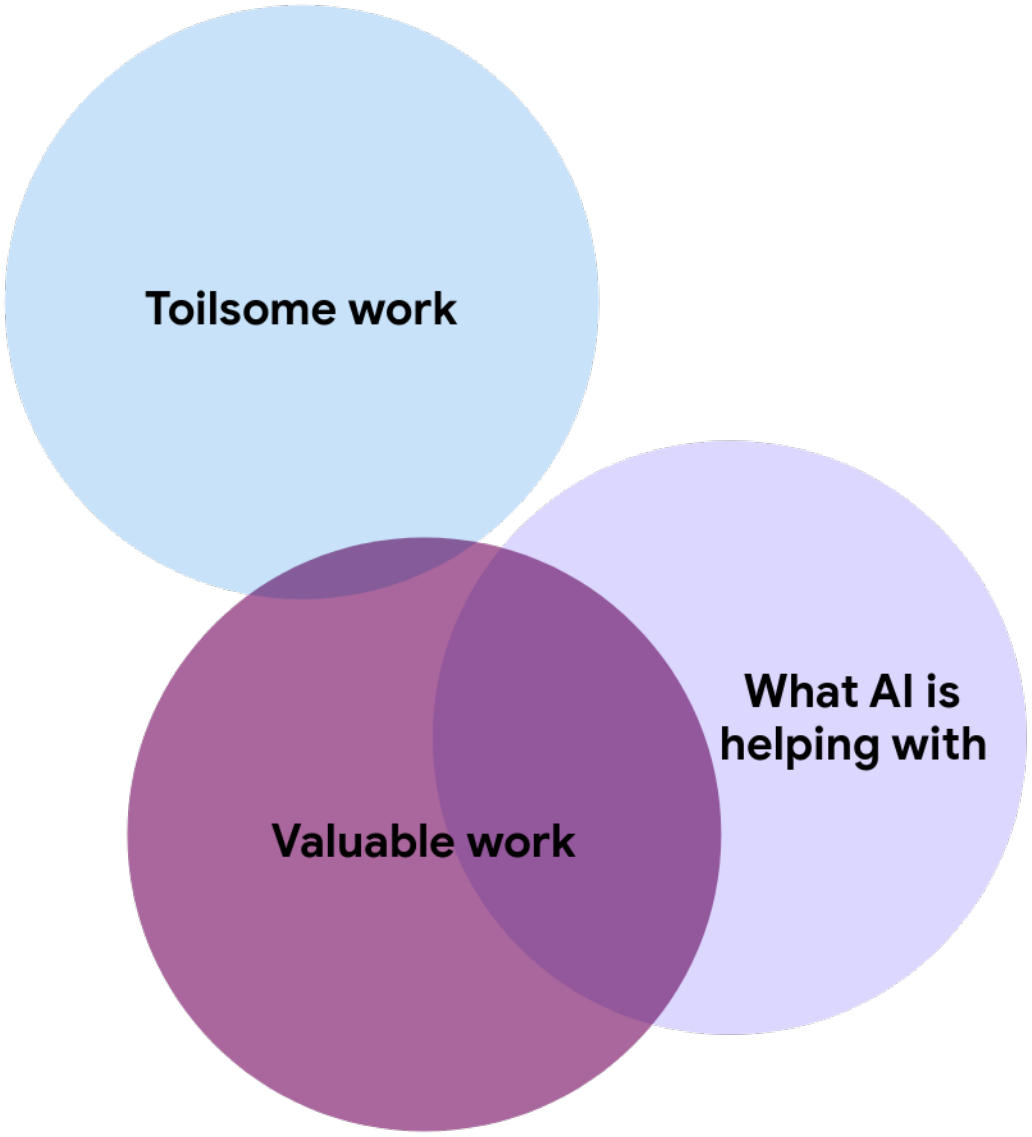


Figure 8: Not data, but a visualization of our hypothesis: AI is helping with our valuable work, but not helping us with our toil.

The promising impact of AI on development workflows

The last section explored outcomes focused on the individual. The next set of outcomes shift focus to explore processes, codebases, and team coordination. Here is a list of the outcomes we measured:

Code complexity	The degree to which code's intricacy and sophistication hinders productivity.
Technical debt	The extent to which existing technical debt within the primary application or service has hindered productivity over the past six months.
Code review speed	The average time required to complete a code review for the primary application or service.
Approval speed	The typical duration from proposing a code change to receiving approval for production use in the primary application or service.
Cross-functional team (XFN) coordination	The level of agreement with the statement: "Over the last three months, I have been able to effectively collaborate with cross-functional team members."
Code quality	The level of satisfaction or dissatisfaction with the quality of code underlying the primary service or application in the last six months.
Documentation quality	The perception of internal documentation (manuals, readmes, code comments) in terms of its reliability, findability, updatedness, and ability to provide support.

As before, our goal here is to understand if these aspects seem to vary as a function of adopting AI. Figure 9 is a visualization that shows our best estimates of the change in these outcomes in relation to a 25% increase in AI adoption.

Overall, the patterns here suggest a very compelling story for AI. Here are the substantial results from this section.

A 25% increase in AI adoption is associated with a...

7.5% increase in documentation quality

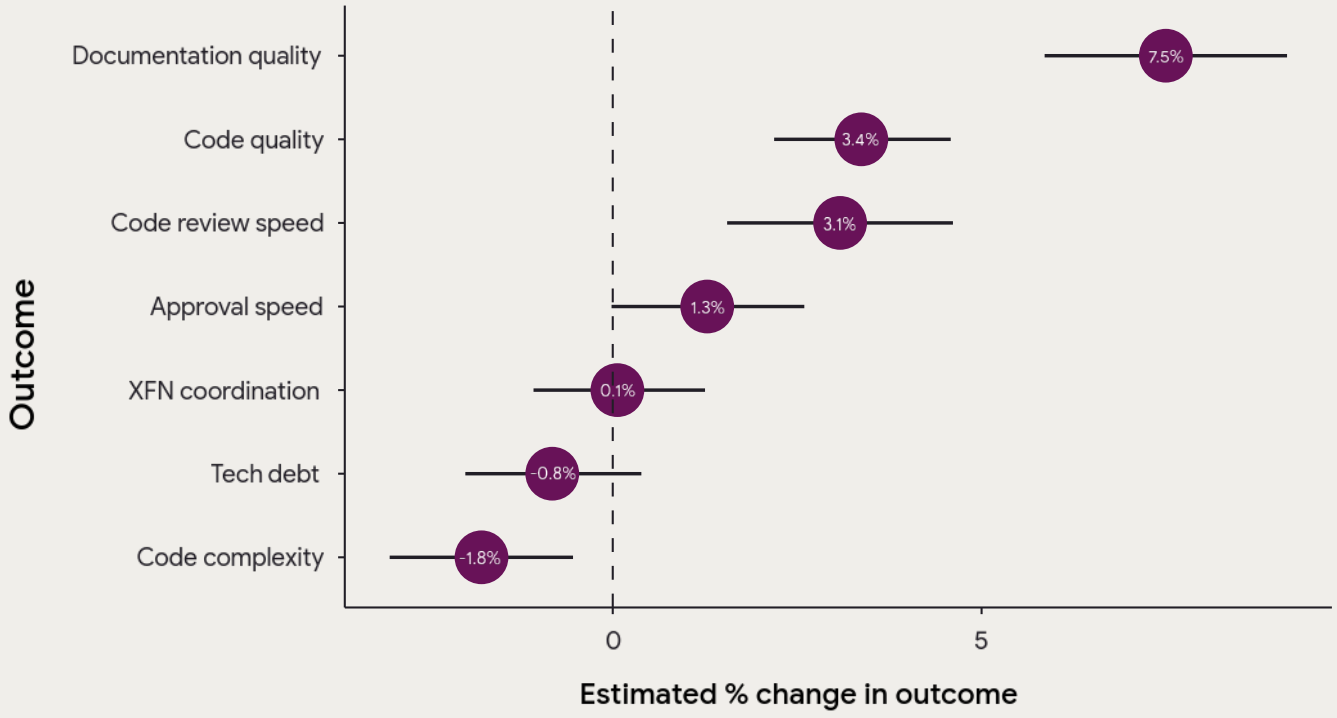
3.4% increase in code quality

3.1% increase in code review speed

1.3% increase in approval speed

1.8% decrease in code complexity

If AI adoption increases by 25%...



Point = estimated value
Error bar = 89% uncertainty interval
Figure 9: Impacts of AI adoption on organizations.

The data presented in the "Artificial intelligence: Adoption and attitudes" chapter show the most common use of AI is for writing code. 67% of respondents report that AI is helping them improve their code. Here, we see further confirmation of that sentiment. AI seems to improve code quality and reduce code complexity (Figure 9). When combined with some potential refactoring of old code, the high-quality, AI-generated code could lead to an overall better codebase. This codebase might be additionally improved by having better access to quality documentation, which people are using AI to generate (see Artificial intelligence: Adoption and attitudes).

Better code is easier to review and approve. Combined with AI-assisted code reviews, we can get faster reviews and approvals, a pattern that has clearly emerged in the data (Figure 9).

Of course, faster code reviews and approvals do not equate to better and more thorough code review processes and approval processes. It is possible that we're gaining speed through an over-reliance on AI for assisting in the process or trusting code generated by AI a bit too much. This finding is not at odds with the patterns in Figure 9, but it also not the obvious conclusion.

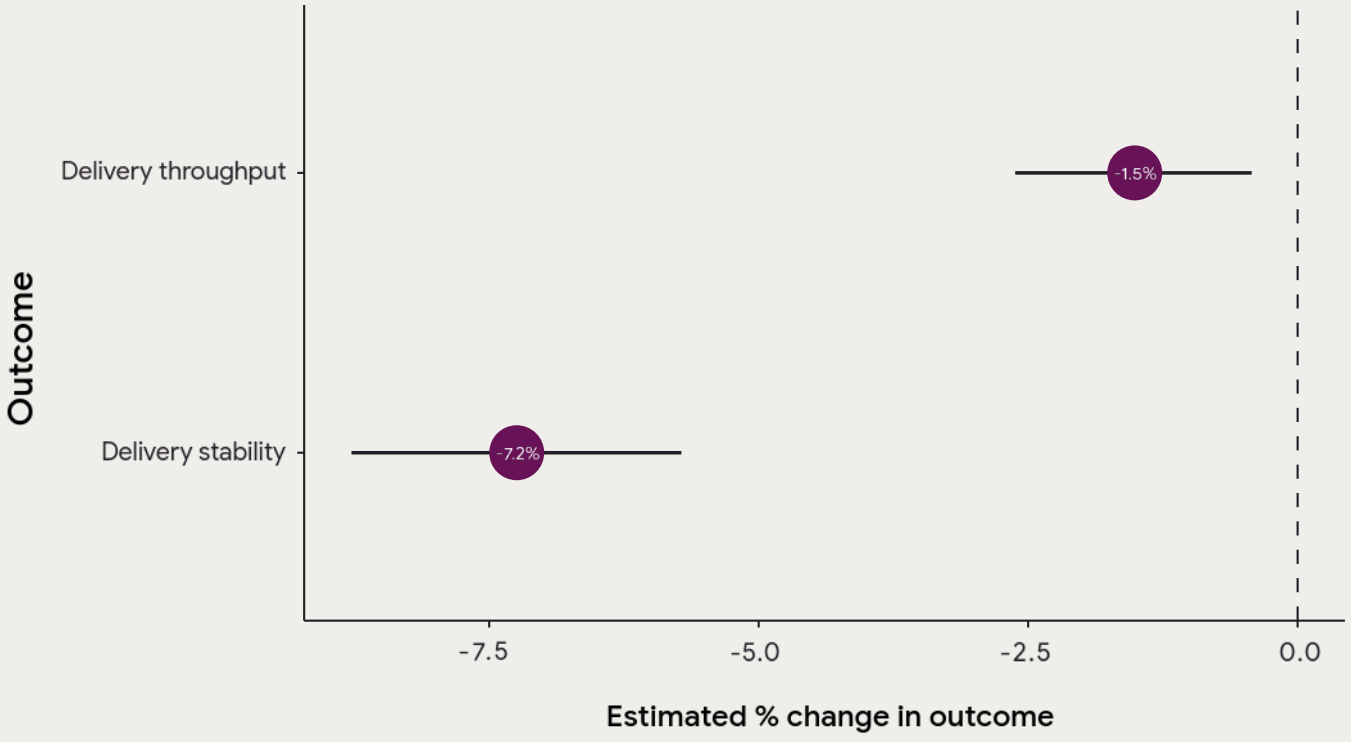
Further, it isn't obvious whether the quality of the code and the quality of the documentation are improving because AI is generating it or if AI has enhanced our ability to get value from what would have otherwise been considered low-quality code and documentation. What if the threshold for what we consider quality code and documentation simply moves down a little bit when we're using AI because AI is powerful enough to help us make sense of it? These two ways of understanding these patterns are not mutually exclusive interpretations; both could be contributing to these patterns.

What seems clear in these patterns is that AI helps people get more from the documents they depend on and the codebases they work on. AI also helps reduce costly bottlenecks in the code review and approval process. What isn't obvious is how exactly AI is doing this and if these benefits lead to further downstream benefits, such as software delivery improvements.

AI is hurting delivery performance

For the past few years, we have seen that software delivery throughput and software delivery stability indicators were starting to show some independence from one another. While the traditional association between throughput and stability has persisted, emerging evidence suggests these factors operate with sufficient independence to warrant separate consideration.

If AI adoption increases by 25%...



Point = estimated value
Error bar = 89% uncertainty interval
Figure 10: Impacts of AI adoption on delivery throughput and stability.

Contrary to our expectations, our findings indicate that AI adoption is negatively impacting software delivery performance. We see that the effect on delivery throughput is small, but likely negative (an estimated 1.5% reduction for every 25% increase in AI adoption). The negative impact on delivery stability is larger (an estimated 7.2% reduction for every 25% increase in AI adoption). This data is visualized in Figure 10.

Historically, our research has found that improvements to the software development process, including improved documentation quality, code quality, code review speed, approval speed, and reduced code complexity lead to improvements in software delivery. So, we were surprised to see AI improve these process measures, while seemingly hurting our performance measures of delivery throughput and stability.

Drawing from our prior years' findings, we hypothesize that the fundamental paradigm shift that AI has produced in terms of respondent productivity and code generation speed may have caused the field to forget one of DORA's most basic principles—the importance of small batch sizes. That is, since AI allows respondents to produce a much greater amount of code in the same amount of time, it is possible, even likely, that changelists are growing in size. DORA has consistently shown that larger changes are slower and more prone to creating instability.

Considered together, our data suggest that improving the development process does not automatically improve software delivery—at least not without proper adherence to the basics of successful software delivery, like small batch sizes and robust testing mechanisms.

The beneficial impact that AI has on many important individual and organizational factors that foster the conditions for high software delivery performance is reason for optimism. But, AI does not appear to be a panacea.

High-performing teams and organizations use AI, but products don't seem to benefit.

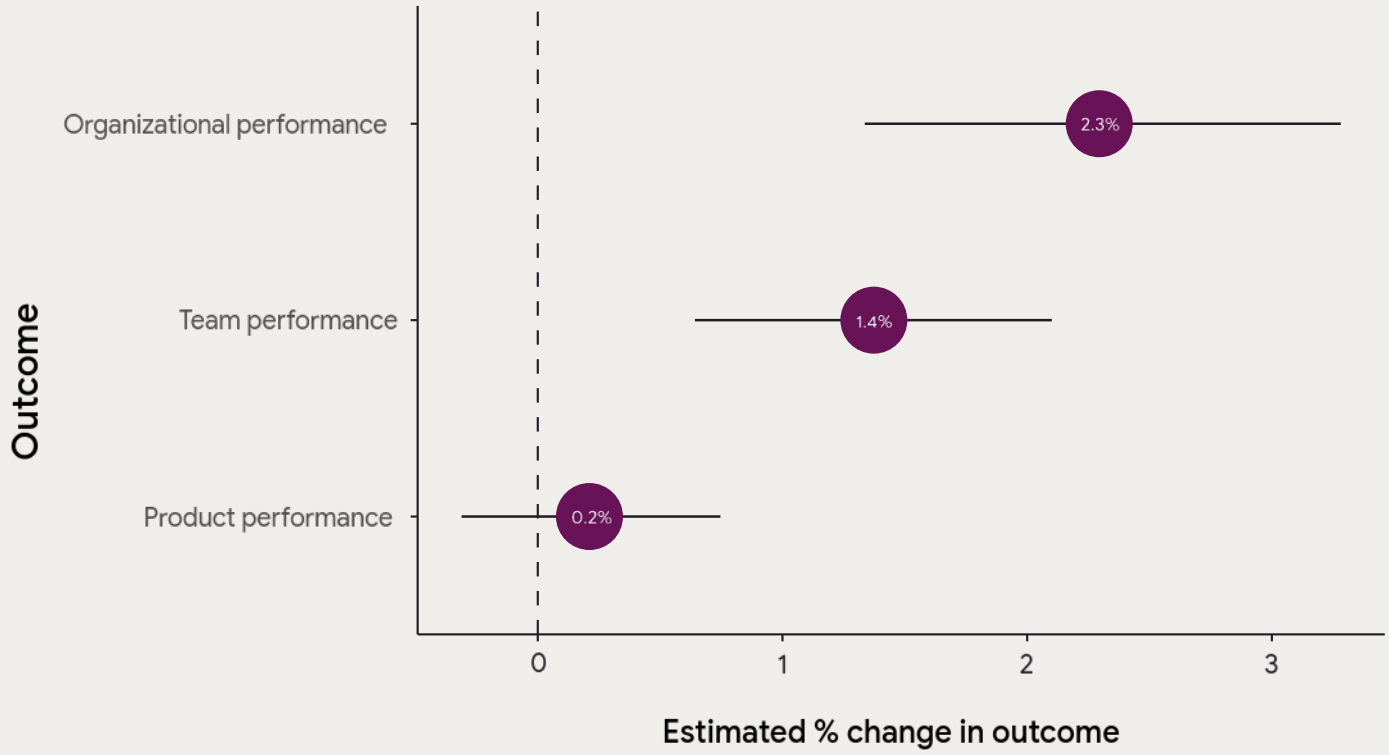
Here we look at AI's relationship with our most downstream outcomes:

Organizational performance	This is a factor score that accounts for an organization's overall performance, profitability, market share, total customers, operating efficiency, customer satisfaction, quality of products/service, and ability to achieve goals.
Team performance	This is a factor score that accounts for a team's ability to collaborate, innovate, work efficiently, rely on each other, and adapt.
Product performance	This is a factor score that accounts for the usability, functionality, value, availability, performance (for example, latency), and security of a product.

Drawing a connection from these outcomes to an individual adopting AI is difficult and noisy. Sometimes it feels like we're trying to analyze the impact of what you had for lunch today on how well your organization performs this year.

There is a logic to making jumps between the micro-level (for example, an individual) to the macro-level (for example, an organization). We discuss that inferential leap in the Methodology chapter. For now, let's just check out the associations:

If AI adoption increases by 25%...



Point = estimated value
Error bar = 89% uncertainty interval
Figure 11. Impacts of AI adoption on organizational, team, and product performance.

Organization-level performance (an estimated 2.3% increase for every 25% increase in AI adoption) and team-level performance (an estimated 1.4% increase for every 25% increase in AI adoption) seem to benefit from AI adoption (Figure 11). Product performance, however, does not seem to have an obvious association with AI adoption. Now, we can shift to trying to understand what is underlying these effects.

We hypothesize that the factors contributing to strong team and organizational performance differ from those influencing product performance.

Teams and organizations rely heavily on communication, knowledge sharing, decision making, and healthy culture. AI could be alleviating some bottlenecks in those areas, beneficially impacting teams and the organizations.

Product success, however, might involve additional factors. Although good products surely have similar underlying causes as good high performing teams and organizations, there is likely a closer and more direct connection to the development workflow and the software delivery, both of which may be still stabilizing after the introduction of AI.

The unique importance of technical aspects underlying a good product might explain part of it, but there is also an art and empathy underlying a great product. This might be difficult to believe for people who think everything is a problem to be resolved through computation, but certain elements of product development, such as creativity or user experience design, may still (or forever) heavily rely on human intuition and expertise.

The fact remains that organization, team, and product performance are undeniably interconnected. When looking at bivariate correlations (Pearson), we find product performance has a medium positive correlation with both team performance ($r = 0.56$, 95% confidence interval = 0.51 to 0.60) and organizational performance ($r = 0.47$, 95% confidence interval = 0.41 to 0.53).

These outcomes influence each other reciprocally, creating clear interdependencies. High-performing teams tend to develop better products, but inheriting a subpar product can hinder their success. Similarly, high-performing organizations foster high-performing teams through resources and processes, but organizational struggles can stifle team performance. Therefore, if AI adoption significantly benefits teams and organizations, it's reasonable to expect benefits for products to emerge as well.

The adoption of AI is just starting. Some benefits and detriments may take time to materialize, either due to the inherent nature of AI's impact or the learning curve associated with its effective utilization.

Perhaps the story is simply that we're figuring out how AI can help organizations and teams before we've fully realized its potential for product innovation and development. Figure 12 tries to visualize how this might be unfolding.

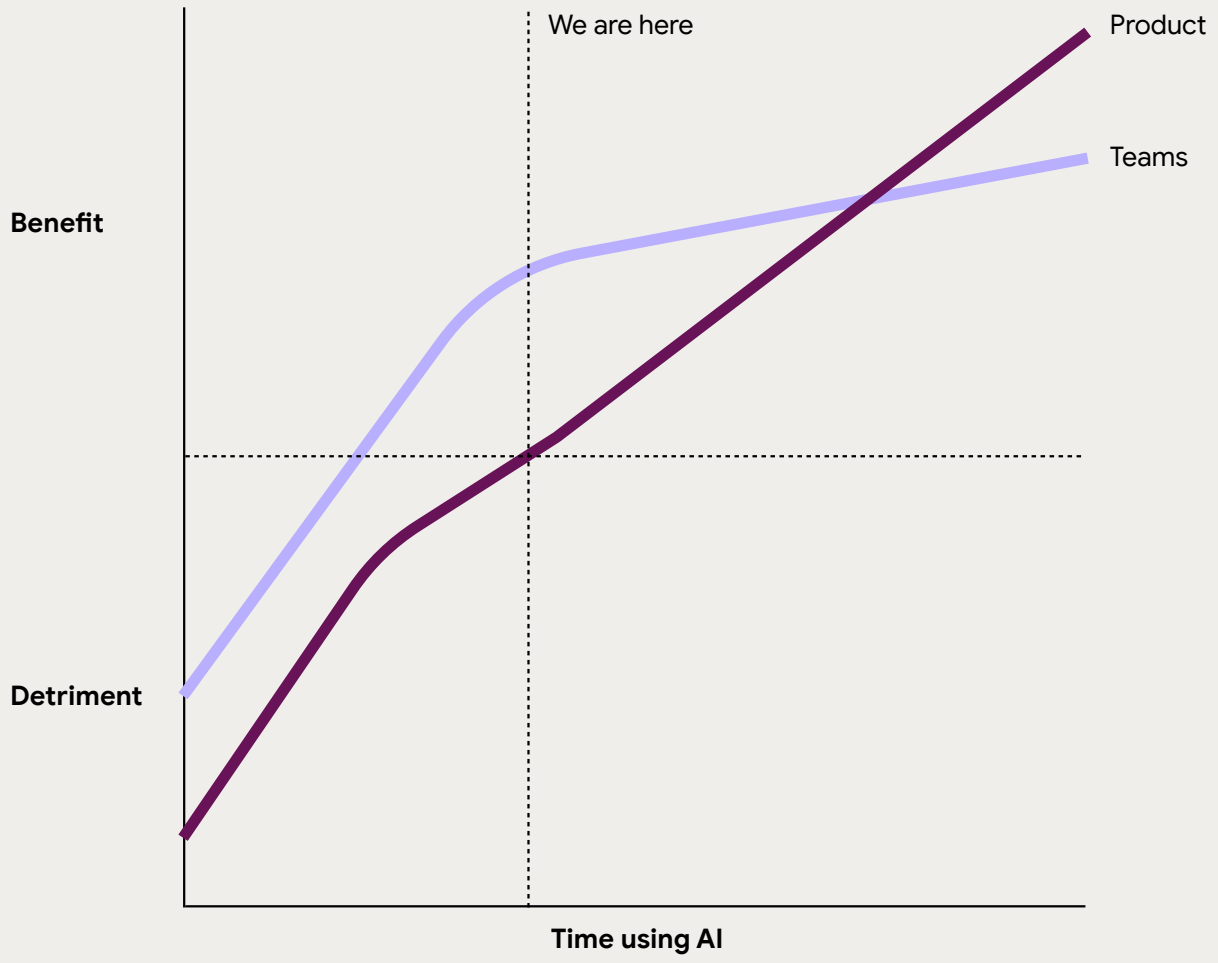


Figure 12: Representations of different learning curves. This is an abstraction for demonstrative purposes. This is not derived from real data.

So now what?

We wanted to understand the potential of AI as it currently stands to help individuals, teams, and organizations. The patterns that are emerging underscore that it isn't all hot air; there really is something happening.

There is clear evidence in favor of adopting AI. That said, it is also abundantly clear that there are plenty of potential roadblocks, growing pains, and ways AI might have deleterious effects.

Adopting AI at scale might not be as easy as pressing play. A measured, transparent, and adaptable strategy has the potential to lead to substantial benefits. This strategy is going to need to be co-developed by leaders, teams, organizations, researchers, and those developing AI.

Leaders and organizations need to find ways to prioritize adoption in the areas that will best support their employees.

Here are some thoughts about how to orient your AI adoption strategy:

Define a clear AI mission and policies to empower your organization and team.

Provide employees with transparent information about your AI mission, goals, and AI adoption plan. By articulating both the overarching vision and specific policies — addressing procedural concerns such as permitted code placement and available tools — you can alleviate apprehension and position AI as a means to help everyone focus on more valuable, fulfilling, and creative work.

Create a culture of continuous learning and experimentation with AI.

Foster an environment that encourages continuous exploration of AI tools by dedicating time for individuals and teams to discover beneficial use cases and granting them autonomy to chart their own course. Build trust with AI technologies through hands-on experience in sandbox or low-risk environments. Consider further mitigating risks by focusing on developing robust test automation. Implement a measurement framework that evaluates AI not by sheer adoption but by meaningful downstream impacts — how it helps employees thrive, benefits those who rely on your products, and unlocks team potential.

Recognize and leverage AI's trade-offs for competitive advantage.

By acknowledging potential drawbacks — such as reduced time spent on valuable work, over-reliance on AI, the potential for benefits gained in one area leading to challenges in another, and impacts on software delivery stability and throughput — you can identify opportunities to avoid pitfalls and positively shape AI's trajectory at your organization, on your team. Developing an understanding not only of how AI can be beneficial, but of how it can be detrimental allows you to expedite learning curves, support exploration, and translate your learnings into action and a real competitive advantage.

It is obvious that there is a lot to be excited about and even more to learn. DORA will stay tuned in and do our best to offer honest, accurate, and useful perspectives, just as it has over the past decade.

1. https://www.goldmansachs.com/insights/top-of-mind/gen-ai-too-much-spend-too-little-benefit
2. https://www.goldmansachs.com/insights/articles/AI-poised-to-drive-160-increase-in-power-demand
3. https://www.washington.edu/news/2023/07/27/how-much-energy-does-chatgpt-use/
4. https://www.gatesnotes.com/The-Age-of-AI-Has-Begun
5. https://www.businessinsider.com/ai-chatgpt-homework-cheating-machine-sam-altman-openai-2024-8
6. https://www.safe.ai/work/statement-on-ai-risk
7. https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/
8. https://www.gitclear.com/coding_on_copilot_data_shows_ais_downward_pressure_on_code_quality
9. https://www.nytimes.com/2024/04/15/technology/ai-models-measurement.html
10. https://dora.dev/capabilities
11. we should be clear that this isn't a unique approach, but it is a somewhat unique approach for this space
12. (P[N]), for example (P1), indicates pseudonym of interview participants.

Platform engineering

Introduction

Platform engineering is an emerging engineering discipline that has been gaining interest and momentum across the industry. Industry leaders such as Spotify and Netflix, and books such as *Team Topologies*[1] have helped excite audiences.

Platform engineering is a sociotechnical discipline where engineers focus on the intersection of social interactions between different teams and the technical aspects of automation, self-service, and repeatability of processes. The concepts behind platform engineering have been studied for many years, including by DORA.

Generally, our research is focused on how we deliver a software to external users, whereas the output of platform teams is typically an inwardly-focused set of APIs, tools, and services designed to support the software development and operations lifecycle.

In platform engineering, a lot of energy and focus is spent on improving the developer experience by building golden paths, which are highly-automated, self-service workflows that users of the platform use when interacting with resources required to deliver and operate applications. Their purpose is to abstract away the complexities of building and delivering software such that the developer only needs to worry about their code.

Some examples of the tasks automated through golden paths include new application provisioning, database provisioning, schema management, test execution, build and deployment infrastructure provisioning, and DNS management.

Concepts in platform engineering such as moving a capability down (sometimes called “shifting down”)[2] into a shared system can seem counter to approaches like 'you build it, you run it.' However, we think of platform engineering as a method to scale the adoption of these practices across an organization because once a capability is in the platform, teams essentially get it for free through adoption of the platform.

For example, if the platform has the capability to execute unit tests and report back results directly to development teams, but without that team needing to build and manage the testing execution environment, then the continuous integration platform feature enables teams to focus on writing high-quality tests. In this example, the continuous integration feature can scale across the larger organization and make it easier for multiple teams to improve their capabilities with continuous testing[3] and test automation.[4]