DORA

Impact of Generative AI in Software Development

Google Cloud

Contents

The AI imperative and the DORA approach 3

Understanding the landscape: The impact of AI 5

How gen AI affects the value of development work 15

Fostering developers' trust in gen AI 23

Helping developers adopt generative AI: Four practical strategies for organizations 30

Measuring success: Key metrics and feedback loops 38

A framework for gen AI adoption 47

Moving forward: Actionable takeaways and the future of AI in development 51

The AI imperative and the DORA approach

Generative AI is driving a profound transformation in the world of software development. The 2024 DORA report[1] confirms this, revealing that 89% of organizations are prioritizing the integration of AI into their applications, and 76% of technologists are already relying on AI in some part of their daily work.

This "AI moment" is fueled by significant investments as leading tech giants are projected to invest approximately $1 trillion in AI development over the next five years. While it's true that the competitive pressure to adopt and use AI is undeniable, we see a more nuanced picture.

Beyond hype and headlines, legitimate concerns about the use of AI in software development have grown with its immense potential. Developers express anxieties about job displacement, security risks, and the fear that AI will diminish the time they spend on work they find truly valuable.

This guide aims to move beyond simple adoption to responsible and effective AI integration throughout the software development lifecycle, maximizing the benefits while mitigating the risks.

For more than a decade, DORA research has focused on identifying the capabilities that drive high-performing software delivery and operations teams. Our findings consistently highlight the importance of continuous improvement, user focus, and data-driven decision-making.

These principles are just as relevant – perhaps even more so – in the age of AI. AI is a powerful tool, but its success depends on how it's implemented.

This report builds on and extends DORA's research published in the Accelerate State of DevOps Report.[2] We review the current landscape of AI adoption, look into its impact on developers and organizations, and outline a framework and practical guidance for successful integration, measurement, and continuous improvement.

Remember that the adoption of new technology can lead to some near-term decreases in productivity and impact. Prioritizing sustainable, long-term improvement can help realize the benefits of adopting generative AI.

1. Accelerate State of DevOps Report 2024 - https://dora.dev/dora-report-2024
2. Ibid.

Understanding the landscape: The impact of AI

Generative AI is already having a significant impact on the daily work lives of practitioners.

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.

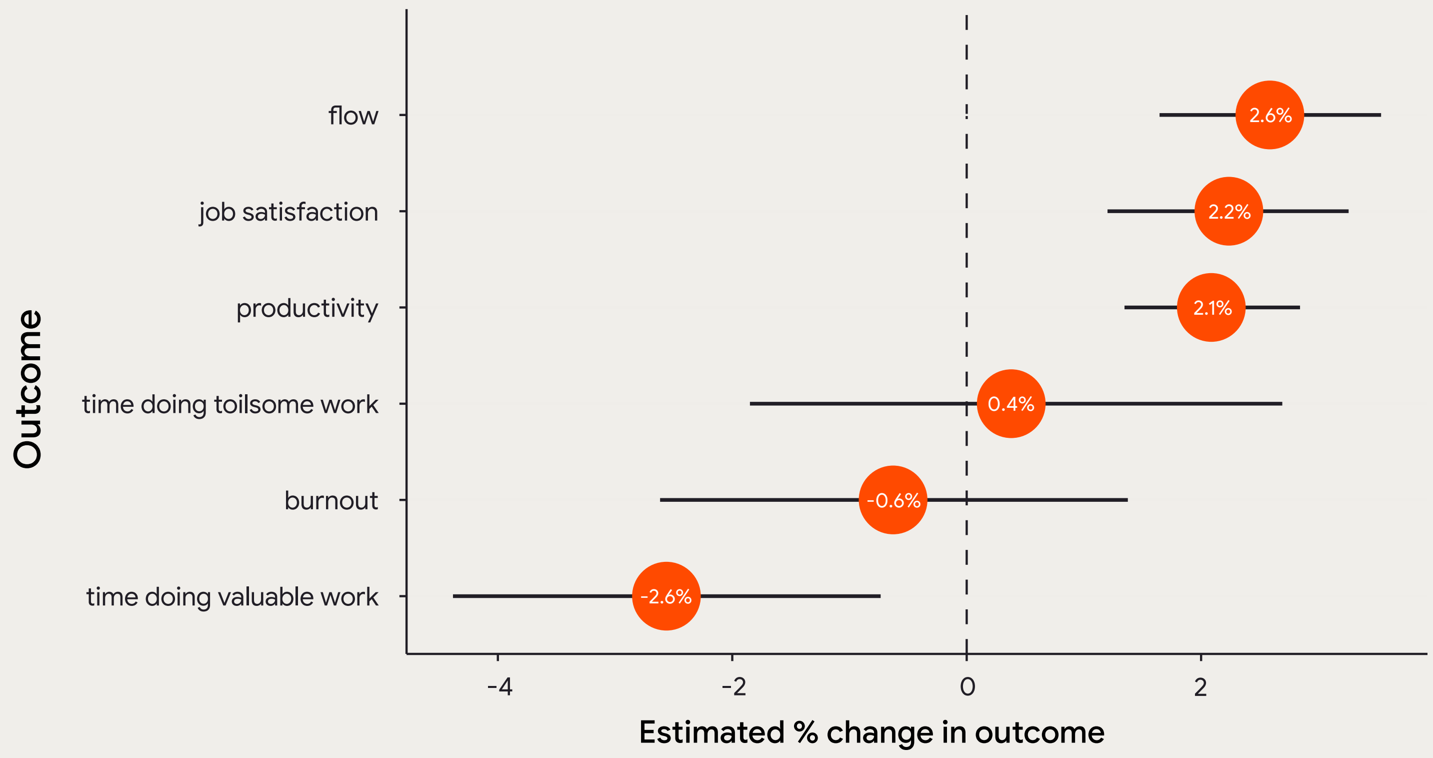
Findings

Practitioner perspectives

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

If AI adoption increases by 25%...



Point = estimated value
Error bar = 89% uncertainty interval
Figure 1: 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 1).

Productivity, for example, is likely to increase by approximately 2.1% when an individual's AI adoption is increased by 25% (see Figure 1). 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.

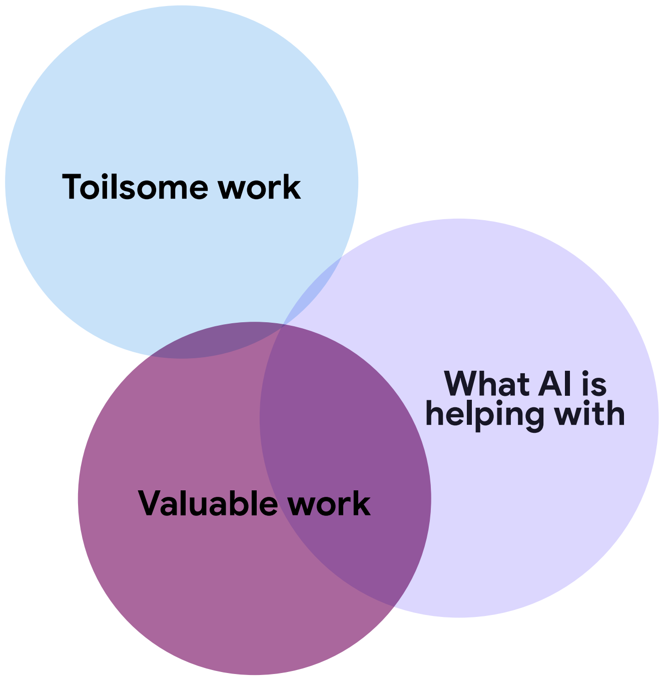
We call our hypothesis the vacuum hypothesis. By increasing productivity and flow, AI is helping people work more efficiently. This efficiency is helping people finish up work they consider valuable faster.

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

To make sense of these counterintuitive findings we explored more deeply what types of work respondents judge to be valuable or toilsome. See the "How gen AI affects the value of development work" chapter for our findings.

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 1).

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



AI is changing the nature of development work.

- Learning and Skill Shifts: AI presents an opportunity for developers to learn new skills, such as prompt engineering and effectively collaborating with AI-powered tools. It's a chance to adapt to evolving industry standards and move beyond routine tasks to higher-level problem-solving.

- Trust and Control: DORA's research indicates developers don't need to absolutely trust AI. Similar to using resources such as Stack Overflow, they view it as a tool that requires verification, critical evaluation, and refinement. It is important that developers maintain control over when and how AI is used in their workflows. See the "Fostering developers' trust in generative artificial intelligence" chapter for more on trust.

Organizational perspectives

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:

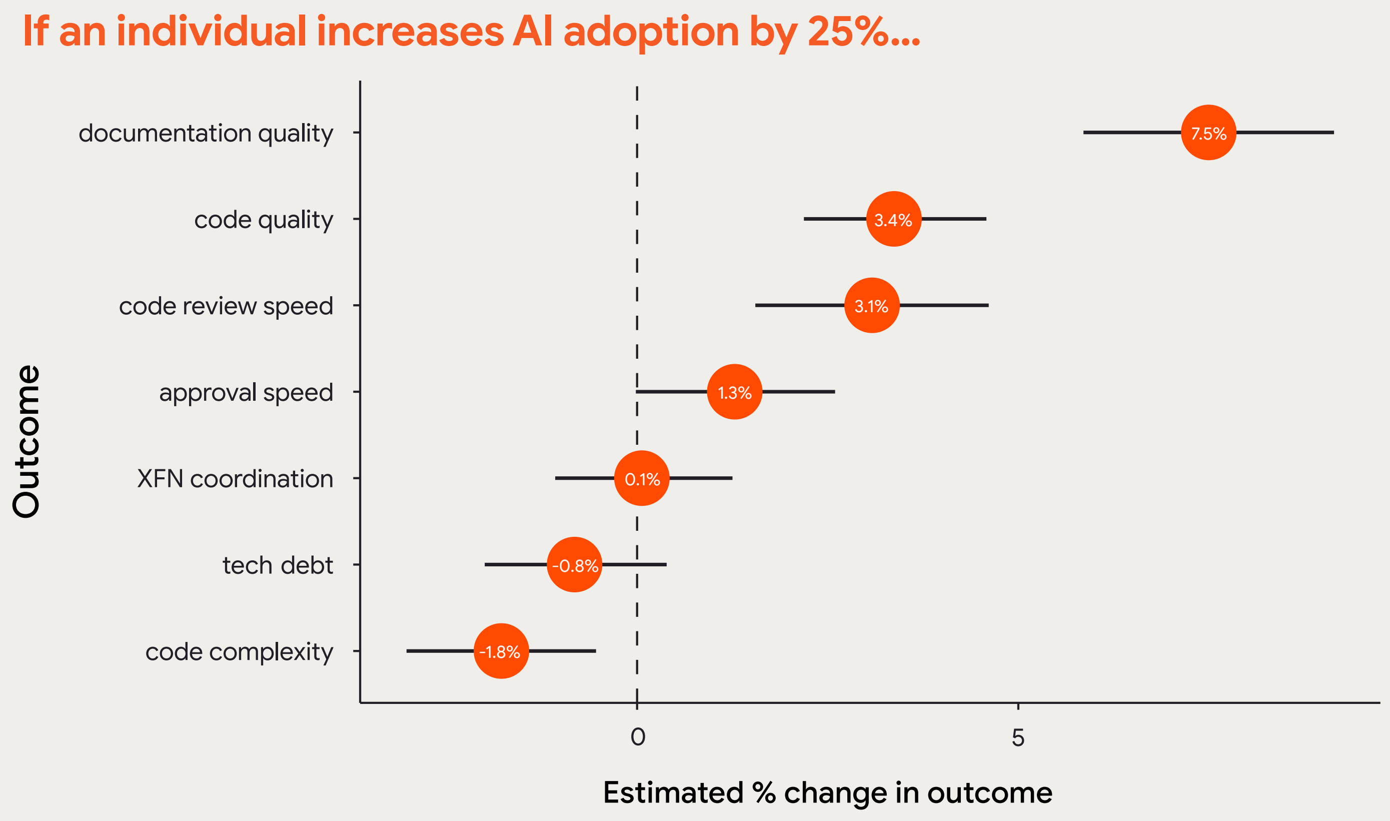
Code complexity	The degree to which code's intricacy and sophistication hinders productivity.
Technical debt	The extent to which existing technical debt within the primary application or service has hindered productivity over the past six months.
Code review speed	The average time required to complete 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 2 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



Point = estimated value
Error bar = 89% uncertainty interval
Figure 2: Impacts of AI adoption on organizations.

The 2024 DORA Report found that 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 2). 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.

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 2).

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 2, but it also not the obvious conclusion.

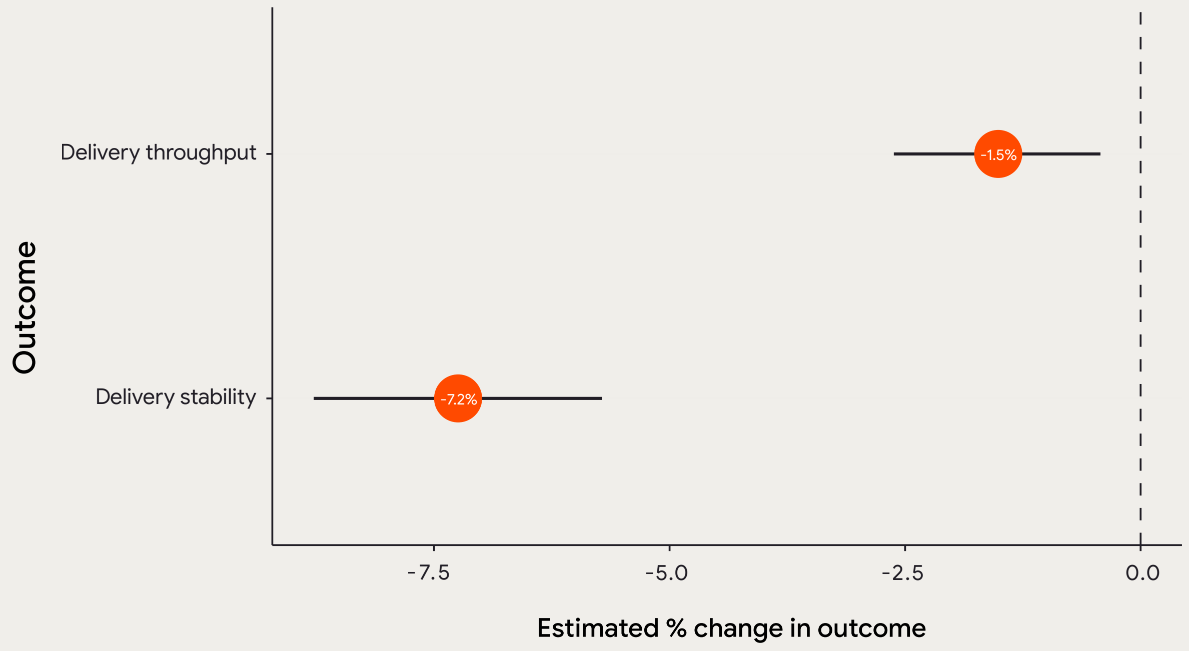
Further, it isn't obvious whether the quality of the code and the quality of the documentation are improving because AI is generating it or if AI has enhanced our ability to get value from what would have otherwise been considered low-quality code and documentation. What if the threshold for what we 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 3: 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 3.

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.

---

[1] Accelerate State of DevOps Report 2024 - https://dora.dev/dora-report-2024

How gen AI affects the value of development work

Developers' well-being and job satisfaction have been a central topic of inquiry for the DORA research team[1] for many years, including as part of the 2024 DORA Report.[2] This year, we were especially interested in investigating whether there is evidence that developers' well-being is affected by the rapid adoption and use of generative artificial intelligence (gen AI) in development work.

We found that developers who more extensively use gen AI in their work report:

- More time in a flow state,
- Higher overall job satisfaction,
- Increased productivity, and
- Less burnout.

Conversely, we found that developers who more extensively use gen AI in their work report:

- No difference in time spent on toilsome work, and
- Less time spent on valuable work.
- These findings were quite surprising for us, for two reasons.

First, spending less time on valuable work when you use gen AI is a notable counterpoint to a popular argument for adopting gen AI in development work—that gen AI will speed up routine development tasks, thereby allowing developers to work on "something better."

Second, as researchers, we found it paradoxical that developers spending less time on work they find valuable would also report more time in flow, more satisfaction, more productivity, and less burnout. It is reasonable to expect that spending one's time on work that is valuable would be a predictor of well-being.

After debating this paradox for some time, and generating several plausible hypotheses to explain it, we realized that we were, in fact, grappling with a more fundamental question:

How do developers actually ascribe "value" to their work?

So, in September 2024, we conducted a series of 10 in-depth interviews with development professionals, each lasting approximately 90 minutes. Participants were asked about the types of development tasks that they found valuable and nonvaluable, how they made assessments about whether specific tasks had value, and whether using gen AI to perform those tasks changed, or would change, their evaluation.

Five views of “value” in development work

Across these interviews, developers described the value of their work using five main views of “value,” from which a framework was derived for understanding the value of development work and gen AI’s impacts on those facets of value. This framework is visualized in Table 1 and further described below.

Five views of “value” in development work

Value	**Utilitarian value**	**Reputational value**	**Economic value**	**Intrinsic value**	**Hedonistic value**
Meaning	“My work impacts the world.”	“I am recognized for my work”	“I am paid for my work.”	“My work is worthwhile.”	“My work is fun and enjoyable.”
Positive impacts of gen AI	Increases productivity	Increases productivity	Increases productivity		Using gen AI can be fun
Negative impacts of gen AI		Complicates ownership of labor	Makes development easier		Gen AI might make some fun tasks obsolete
Neutral impacts of gen AI				Changes educational standards	Using gen AI for fun tasks is optional

Table 1: The five views of "value" of development work described by participants, their meanings, and the anticipated impacts of gen AI on each of these perspectives.

Utilitarian value describes the impact that the work has on the world.

- Utilitarian value is determined by the impact that developers perceive their product having in the world and the impact of specific development tasks on the creation of that product.
- Gen AI can be anticipated to have a potentially positive impact on the utilitarian value of development work, by increasing the pace at which developers work.

Reputational value describes the recognition that an individual performed work.

- Reputational value is determined by the utilitarian value of that work and whether its performance can be ascribed to/owned by any individual developer.
- Gen AI can be anticipated to have a potentially positive impact on the reputational value of development work, by increasing the utilitarian value of an individual developer's labor.
- However, gen AI can also be anticipated to have a potentially negative impact on the reputational value of development work, by calling into question the ownership of the labor performed with gen AI.

Economic value describes the pay associated with performing development work.

- Economic value is determined by the utilitarian value of the work, the Reputational value of the work, and the (in)accessibility of the skillset, where niche skillsets are more highly valued.
- Gen AI can be anticipated to have a potentially positive impact on the economic value of development work, by increasing the utilitarian value of an individual developer's labor.
- However, gen AI can also be anticipated to have a potentially negative impact on the economic value of development work, by reducing the reputational value of development work and by making the requisite skillset more accessible.

Intrinsic value describes the inherent worthwhileness of performing development work.

- Intrinsic value is highly subjective, deeply personal, and often argued from a perspective of valuing traditional ways of learning and achieving.

- Gen AI can be anticipated to have a neutral impact on the intrinsic value of development work, because historical examples suggest novel technologies change norms around educational and knowledge standards (consider wayfinding pre-GPS or mental math pre-mobile phone calculator).

Hedonistic value describes the enjoyment resultant from performing development work.

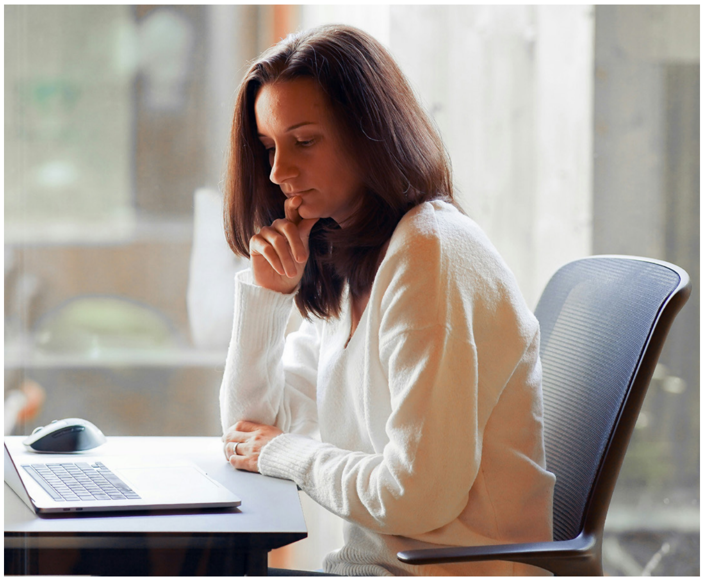
- Hedonistic value is highly subjective and dependent upon the personal interests of the developer in question.

- Gen AI can be anticipated to have a potentially positive impact on the hedonistic value of development work, if the developer enjoys using gen AI or gen AI makes enjoyable tasks accessible.

- Gen AI can also be anticipated to have a potentially negative impact on the hedonistic value of development work, if the developer enjoys performing a task gen AI renders obsolete.

- Gen AI can also be anticipated to have a neutral impact on the hedonistic value of development work, because, for now, developers are not being forced to use gen AI and so can perform any enjoyable task if they want.

Put simply, gen AI has variable and interconnected impacts on how developers, and the world, perceive the value of development work.



Five strategies for ensuring AI amplifies developers' value

Allow use of gen AI throughout the software development and delivery process.

Developers who prioritized the utilitarian value of their work—emphasizing their individual impact on the world—perceived gen AI as a clear value-booster. That is, by increasing the speed at which developers can change an application, the time taken to produce an impact in the world is reduced. So, allowing the use of gen AI in as many development tasks as possible may increase the perceived utilitarian value of development work by expediting the completion of more steps throughout the development process.

Acknowledge the labor of working with gen AI.

Developers who prioritized the reputational value of their work—emphasizing the benefits of being recognized for having completed work—perceived gen AI as a potential value-reducer. That is, by supporting developers in completing their work, it is possible that gen AI will be given the credit for the work performed, at the expense of the developer. Although gen AI can scaffold many development tasks, working with gen AI still requires labor on the part of the developer. There are even new job titles, like prompt engineer, and programming paradigms, like chat-oriented programming,[3] emerging in response to the labor of using gen AI in development. Leadership explicitly acknowledging this labor involved in working with gen AI may help assuage fears about gen AI's impacts on the reputational value of development, by centering developers' contributions to AI-assisted development work.

Reward work outcomes, not just time spent.

Developers who prioritized the economic value of their work—emphasizing the pay associated with their labor—had mixed feelings about gen AI's potential impacts on development work's value. On one hand, they perceived gen AI to increase their productivity, thereby making their time more economically valuable. On the other hand, because gen AI expedites task completion, some developers expressed concerns about whether increases in individual productivity would cause reductions in workforce or paid hours. How gen AI will shape our future global economic systems has been widely debated and would be impossible for us to predict.

But, we believe that rewarding developers for the outcomes of their work, not just how much time they spend in the office, may help mitigate these fears and encourage gen AI use.

Frame gen AI as an opportunity to learn new skills

Developers who prioritized the intrinsic value of their work—emphasizing the inherent value of understanding the skills of development—gen AI was perceived as a potential value-reducer. That is, gen AI use might make it less important for developers to maintain some development skills. However, as noted above, there are myriad examples throughout history of novel technologies which shifted the norms of educational standards. So, gen AI might also make other skills, like prompt engineering, more important for developers to hone. Framing gen AI as an opportunity to learn new skills across the software development lifecycle may help assuage developers' fears about other skills being obsolesced, by emphasizing that novel technologies historically change what is important to know, rather than making knowing things unimportant.

Support developers in choosing not to use gen AI for tasks they enjoy.

Developers who prioritized the hedonistic value of their work—emphasizing the pleasure derived from performing certain tasks—gen AI was perceived as a potential value-reducer. That is, because gen AI is capable of doing many tasks that these developers enjoy, it might spoil some of the fun of development. However, we are not aware of any case where gen AI use is compulsory. Although developers report many benefits of gen AI use in their work, we believe that organizations supporting developers who would prefer to perform development tasks without AI assistance is important for preserving the joy of development work that enticed many professional developers into the field initially.

Conclusion

As gen AI becomes increasingly central to developers' everyday worklives, it is important to understand how its use impacts the way developers understand and perceive the value of their professional labor. For many people, the value of their professional labor extends well beyond the economic value placed on it and the associated compensation to encompass aspects of personal identity, pleasure, and contribution to the world. After working with developers for more than a decade, the DORA team was not surprised to find that developers understand the value of their work as a complicated and nuanced web of factors bridging utilitarian, reputational, economic, intrinsic, and hedonistic perspectives on the meaning of "value." Since this is a complicated topic and gen AI's role in the valuation of development work is still evolving, it will be important to build upon and refine this research over time. But, our desire is that this framework and set of recommendations for organizations will help us, collectively, continue to identify opportunities for gen AI to maximize developers' productivity and well-being. We look forward to working with developers, organizations, and the DORA Community[4] toward this goal.

1. DORA's Research Team - https://dora.dev/research/team/
2. Accelerate State of DevOps Report 2024 - https://dora.dev/dora-report-2024
3. Chat-oriented programming (CHOP) in action - https://sourcegraph.com/blog/chat-oriented-programming-in-action
4. DORA Community - https://dora.community

Fostering developers' trust in gen AI

It’s no secret that generative artificial intelligence (gen AI) is rapidly changing the landscape of software development, with discussions about best practices for applying this transformative technology dominating the popular press. Perhaps nowhere on Earth have these discussions been more frequent and passionate than inside the organizations dedicated to making gen AI accessible and useful to developers, including at Google. During one such discussion between researchers on Google Cloud's DORA[1] and Engineering Productivity Research (EPR) teams, we were struck by a recurring finding common to development professionals both inside and outside of Google:

Using gen AI makes developers feel more productive, and developers who trust gen AI use it more.

On the surface, this finding may seem somewhat... obvious. But, for us, it highlighted the deep need to better understand the factors that impact developers' trust in gen AI systems and ways to foster that trust, so that developers and development firms can yield the most benefit from their investment in gen AI development tools.

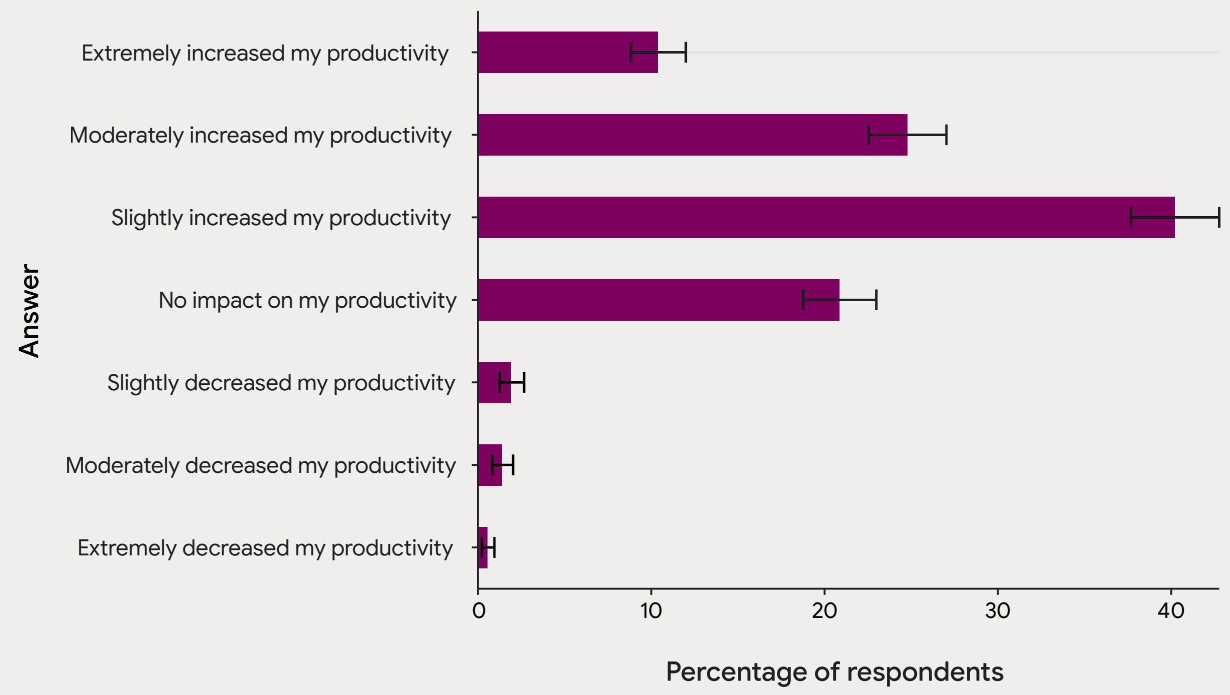
Here, we reflect on findings from several studies conducted at Google,[2] regarding the productivity gains of gen AI use in development, the impacts of developers' trust on gen AI use, and the factors we've observed which positively impact developers' trust in gen AI. We conclude with five suggested strategies that organizations engaged in software development might employ to foster their developers' trust in gen AI, thereby increasing their gen AI use and maximizing gen AI-related productivity gains.

Trust and productivity

Research conducted by Google and other respected voices in technology research has found consistently high use of gen AI by software developers, and developers hold largely favorable views of using gen AI tools at work. Our research suggests that this warm reception of gen AI amongst professional developers is due, in no small part, to significant increases in reported productivity from using gen AI.

The DORA team found that, outside of Google, 75% of 2024 DORA survey respondents reported positive impacts of gen AI on their productivity. Internal to Google, the EPR team found a similar number of developers reported a positive impact of gen AI on their productivity, as well.

Perceptions of productivity changes due to AI



Error bar represents 89% uncertainty level

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

Importantly, developers who trust gen AI more reap more positive productivity benefits from its use. In a logs-based exploration of Google developers' trust in AI code completion, our EPR team found that developers who frequently accepted suggestions from a gen AI-assisted coding tool submitted more change lists (CLs) and spent less time seeking information than developers who infrequently accepted suggestions from the same tool. This was true even when controlling for confounding factors, including job level, tenure, development type, programming language, and CL count.[3] Put simply, developers who trust gen AI more are more productive.

Unfortunately, developers' trust in gen AI appears to be relatively low. According to StackOverflow's 2024 Developer Survey,[2] trusting the output of gen AI is presently developers' number one challenge with AI at work, and Google's research supports this finding. The DORA team found 39% of developers outside of Google trust the quality of gen AI output only "a little" or "not at all."

We find these low levels of trust concerning, because they suggest that a subset of developers may not be experiencing the productivity gains they could from gen AI.

Five strategies for fostering developers' trust in gen AI

Given the significant investment many organizations are making in gen AI, and the likelihood that increasing developers' trust in gen AI would improve returns on those investments, we believe it is important to reflect on our prior research and offer the following five strategies organizations might take to foster developers' trust in gen AI:

Establish a policy about acceptable gen AI use, even if your developers are good corporate citizens.

Policies about the tasks, purposes, and data that developers can and cannot use in conjunction with gen AI tools are often viewed through the lens of preventing "bad" behavior, in the form of proprietary data leaks, security breaches, and poor stewardship of user data. While important, preventing irresponsible use of gen AI is only half of the story. Our data suggest establishing clear guidelines encouraging acceptable use of gen AI will likely also promote cautious and responsible developers to use gen AI, by assuaging fears of unknowingly acting irresponsibly. In our survey, the DORA team found that employees who felt their organizations were more transparent about the use of gen AI trusted gen AI more.

Unfortunately, in qualitative interviews, most participants were unable to say with certainty whether their organization had any policy about the use of gen AI in development, at all. Together, these findings suggest that many firms could increase their employees' trust in gen AI by providing more explicit direction about its use and sufficiently advertising that policy to their workforce. Google has previously published guidance on how to craft an acceptable use policy for gen AI,[3] including recommendations for specifying scope, enforcement responsibilities, and delegating accountability for data governance and information security. Our research suggests that comprehensive acceptable use policies which include these elements can help developers feel empowered to rely on and use gen AI, by knowing that appropriate risk-mitigating guardrails are in place. But, we believe that even a lightweight policy, such as a guide to aligning gen AI-produced code to the company's programming conventions, is likely a better signal that gen AI use is sanctioned than a simple lack of explicit prohibitions. For some firms, it may even be beneficial to externalize company policies about gen AI use, to assure end users of their products that development was handled responsibly and using reliable sources.[4]

Double-down on fast high-quality feedback, like code reviews and automated testing, using gen AI as appropriate.

Developers' perceptions that their organization's code review and automated testing processes are rigorous appear to foster trust in gen AI. This is likely because appropriate safeguards assuring them that any errors that may be introduced by gen AI-generated code will be detected before it is deployed to production. Interestingly, data from the DORA team suggests that the adoption of gen AI makes code reviews faster and improves reported code quality, likely by allowing a wider breadth of code to be analyzed at a faster pace than could reasonably be expected of a human. So, it is possible there is a virtuous cycle in which applying appropriate safeguards fosters trust in gen AI and encourages its use in feedback processes, like code reviews and testing. This strengthens the robustness of the safeguards which foster trust, further promoting its use. The logical entry point for this cycle is likely encouraging developers to use gen AI in tasks that feel low-risk, until their trust is built and reinforced over time.

Provide opportunities for developers to gain exposure with gen AI, especially those which support using their preferred programming language.

Trust in gen AI increases as developers gain exposure to it and grow more familiar with its uses, strengths, and limitations. Additionally, developers appear to trust gen AI more when engaging it for tasks in their preferred programming language, likely because they have more expertise to assess and correct its outputs. Providing opportunities to gain exposure to gen AI, like training, unstructured activities, or slack time devoted to trying gen AI, will help increase trust, especially if such activities can be performed in developers' preferred programming language in which they are best equipped to evaluate gen AI's quality.

Encourage gen AI use, but don't force it.

Leadership actively encouraging the use of gen AI in development work appears to be effective in promoting its use amongst individual contributor developers. At the same time, control over the degree to which gen AI intervenes and the tasks in which it is employed increases developers' overall trust in gen AI tools. So, while it is likely to be to the benefit of the organization for people in leadership roles to encourage their ICs to test, evaluate, and employ gen AI in their daily work, it is important developers do not feel obligated to use gen AI. One approach to encouraging gen AI use in a manner that prioritizes developers' sense of control is to promote the spread of knowledge organically, by building community structures[5] to foster conversations about gen AI.

Help developers think beyond automating their day-to-day work and envision what the future of their role might look like.

Fears that gen AI might lead to a future loss of employment for development professionals have been well-publicized and were a recurring theme in our interviews and survey data. We believe much of this fear stems from the fact that the most common uses of gen AI in development replicate the daily work of developers, like writing code or test cases. Delegating these tasks to gen AI has, indeed, made developers more productive. However, without a clear vision for what the transformed role of a developer working at a higher level of abstraction in which these repetitive tasks are delegated to gen AI resembles, it will be hard to assuage fears of unemployment. That is, delegating mundane tasks, such as generating test paths, creating documentation, and providing system health monitoring, is a clear small step toward using gen AI effectively. But, long-term, developers will likely need guidance about how to reallocate the time saved, and encouragement to explore new ways to improve user experience, innovate with emerging technologies, and add value for their companies and users. We are unable to predict what development work will look like in the future, or what new tasks will be performed as a result of gen AI increasing developer capacity. But, we believe that acknowledging gen AI will fundamentally change development work, and empowering developers to have a voice in shaping the future of their profession will foster trust in gen AI by helping developers co-create and move toward a world where gen AI transforms their work, rather than simply *replicating* it.

Conclusion

We are still at a moment in which the ubiquitous use of gen AI in development is nascent, and the long-term efficacy of the strategies proposed above remains to be determined. Yet, given our opportunity to see emerging AI-powered development tools in action at Google and across the industry, we wanted to share our insights and observations to help organizations navigate these still-uncertain waters by sharing insights from our early inquiries. While the long-term future of gen AI use across the software development lifecycle is inevitable, the near-term path to successfully realizing that future is less clear. We believe those individuals and organizations that embrace this change, start with small projects, learn, and iterate will be in a better position to proactively navigate it than those that sit on the sidelines. The good news is that you are not alone in this journey, and we look forward to continuing to co-create the future of software development with you.

1. DORA's Research Team - https://dora.dev/research/team/
2. A. Murillo et al., "Understanding and Designing for Trust in AI-Powered Developer Tooling," in IEEE Software, vol. 41, no. 6, pp. 23-28, Nov.-Dec. 2024, doi: 10.1109/MS.2024.3439108; S. D'Angelo, A. Murillo, S. Chandra and A. Macvean, "What Do Developers Want From AI?," in IEEE Software, vol. 41, no. 3, pp. 11-15, May-June 2024, doi: 10.1109/MS.2024.3363538; ML-Enhanced Code Completion Improves Developer Productivity - https://research.google/blog/ml-enhanced-code-completion-improves-developer-productivity/ AI in software engineering at Google: Progress and the path ahead - https://research.google/blog/ml-enhanced-code-completion-improves-developer-productivity/
3. A. Brown, et al., "Identifying the Factors That Influence Trust in AI Code Completion" in AIware 2024: Proceedings of the 1st ACM International Conference on AI-Powered Software, pp. 1-9, doi: 10.1145/3664646.3664757
4. StackOverflow's 2024 Developer Survey - https://survey.stackoverflow.co/2024/ai#developer-tools
5. How to craft an Acceptable Use Policy for gen AI (and look smart doing it) - https://cloud.google.com/transform/how-to-craft-an-acceptable-use-policy-for-gen-ai-and-look-smart-doing-it
6. How and when Gemini for Google Cloud cites sources - https://cloud.google.com/gemini/docs/discover/works#how-when-gemini-cites-sources
7. How to transform your organization: Build community structures to spread knowledge - https://dora.dev/guides/devops-culture-transform/#build-community-structures-to-spread-knowledge

Helping developers adopt generative AI: Four practical strategies for organizations

Generative AI is quickly becoming ubiquitous to the world of software development. AI is poised to transform every stage of the software development lifecycle – from initial design to ongoing maintenance. We know that AI is already reshaping how technologists work, and fundamentally altering how organizations function. The 2024 DORA report[1] indicates that AI adoption is occurring as a result of both top-down and bottom-up efforts, showing that 89% of organizations are prioritizing the integration of AI into their applications and 76% of technologists report relying on AI for parts of their daily work.

For AI adoption to lead to long-term success, organizations need to address a crucial stumbling block– **scaling its adoption across the entire organization**.

Early successes seen from adopting new technologies often fail to take hold when organizations struggle to scale the technology, leaving teams scrambling to keep up as the full potential of new technologies remains untapped.

However, this doesn’t have to happen with AI. Here are four practical, research-backed strategies that can help your organization move from isolated experiments to sustainable and widespread adoption of AI.

When you look at these strategies, you might notice that there are many ways you could apply them. This might seem daunting, but it’s by design. While our findings offer guidance and specific approaches to follow, the successful implementation of these strategies requires a deep understanding of the organization seeking to use them, and a willingness to experiment and iterate.

1. Share and be transparent about how your organization plans to use AI
2. Address developer concerns about AI's impact
3. Allow ample time for developers to learn how to use AI
4. Create policies that govern the adoption of AI

Ultimately, the goal is to create an environment where developers feel confident and empowered to use AI when it makes the most sense, maximizing its potential benefits while minimizing potential risks.

Each of these strategies assumes that AI tooling is available to practitioners, and that initial experiments have indicated success in driving beneficial outcomes to practitioners and the organization, including improving productivity and job satisfaction.

Share and be transparent about how the organization plans to use AI

Our research suggests that organizations that apply this strategy can gain an estimated 11.4% increase in team adoption of AI compared to companies that do not openly communicate their AI plans to employees. We recommend that organizations provide employees with transparent information about their AI mission, goals, and adoption plan. By articulating both the overarching vision and specific policies, including addressing procedural concerns such as permitted code placement and available tools, you can alleviate employee apprehension and position AI as a means to help everyone focus on more valuable, fulfilling, and creative work.

You'll notice a call towards transparency reappearing in the strategies that follow. Transparency is part of a healthy organizational culture, and over the years DORA has consistently shown it can help teams be more successful at implementing technical capabilities associated with improved outcomes.[2]

Address developer concerns about AI's impact

The rise of AI has also sparked concerns among developers. DORA found that nearly 15% of respondents expect AI to have a detrimental effect in their careers.

This belief might drive some developers to adopt AI not out of genuine interest or recognition of its potential benefits, but rather out of a reactive need to protect their careers.

One developer we spoke to said, "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... no one wants to get left behind."

This reactionary approach, which may be fueled by anxieties about job displacement, can hinder the true potential of AI. However, DORA research indicates that organizational leaders who address these concerns in clear terms can enable developers to focus on learning how to best use AI.

Our findings indicate that organizations that take steps to alleviate developer anxieties are estimated to have 125% more team adoption of AI than those who ignore those concerns.

We recommend that organizations work towards alleviating developer concerns about job displacement by increasing transparency and communication on how they plan to use AI. This approach can help organizations maintain and even enhance a culture of psychological safety, where developers feel supported as they work to embrace new tools.

Allow ample time for developers to learn how to use AI

Integrating AI into developer workflows has a learning curve. Our data indicates that individual reliance on AI peaks at around 15 to 20 months into using the tool, suggesting that learning how to use AI requires dedicated time for experimentation, practice, and integration with existing workflows.

While providing resources including training materials and documentation can be helpful, our findings suggest that actively encouraging developers to integrate AI into their workflows leads to a 27% increase in AI adoption. However, these findings also suggest that simply giving developers dedicated time during work hours to explore and experiment with AI tools leads to a 131% increase in team AI adoption.

We envision this dedicated time to be a combination of reducing productivity expectations during the adoption phase and giving developers ample heads-down time to explore these tools on their own — while also providing them with opportunities to collaborate and learn from others around them. This may take the form of community-led hack-a-thons, lunch-n-learns, or communities of practice.[3]

Expecting developers to acquire these skills in their personal time can lead to frustration, burnout, and ultimately hinder adoption. By providing dedicated time for AI learning and exploration during work hours, organizations demonstrate a commitment to developer growth and create a supportive environment for successful AI adoption.

Providing dedicated time for developers to learn and experiment with AI can create an environment that enables — not forces — AI adoption. The goal is for organizations to foster a healthy culture where developers feel supported in the adoption of AI, without creating top-level mandates that force them into adopting this technology.

While our findings indicate that mandatory training does lead to small increases in AI adoption, we do not believe this to be a sustainable or fruitful strategy. Widespread adoption requires developers to embrace this technology, and to believe its adoption will lead to better outcomes for them and their organizations.

Create policies that govern the adoption of AI

Establishing clear organizational policies around AI usage can provide developers with a framework to confidently, responsibly, and effectively use these tools. Well-defined policies outline appropriate use cases, ethical considerations, and potential risks, guiding developers in making informed decisions about how to best use AI in their workflows.

This clarity can reduce uncertainty, encourage responsible experimentation, and promote a consistent approach to AI adoption across the organization. Our survey suggests that organizations that create AI acceptable-use policies[4] show a 451% increase in AI adoption as compared to companies that don't.

Further, DORA demonstrated that there's a strong association between the quality of internal documentation and AI adoption.[5] Perhaps high quality documentation helps provide the clarity necessary for people to confidently adopt AI.

Addressing security concerns through specific policies and safeguards can help build confidence and mitigate potential risks associated with AI adoption.

By understanding potential risks and current practices for data security and privacy, developers can better discern appropriate use cases for AI, ensuring compliance with organizational policies. This guidance can foster a sense of responsibility and can help developers use AI in a manner that prioritizes security and innovation.

Implementing these guidelines may lead to some near-term decreases in the adoption of generative AI. Our data suggests there is a lot of uncertainty about how guidelines on mitigating security and privacy risks, and guidelines that outline when and where to use AI will impact its adoption. This is likely because the impact of t hese strategies is in the details and depends on the context.

This isn't necessarily a bad thing! It might even be a sign that developers are exercising thoughtful discretion when deciding when to use and not use AI. Identifying use cases where developers are incorporating AI alongside use cases where developers are choosing to not adopt AI may help organizations identify a sustainable, long-term approach that uses ongoing feedback to iterate as necessary.

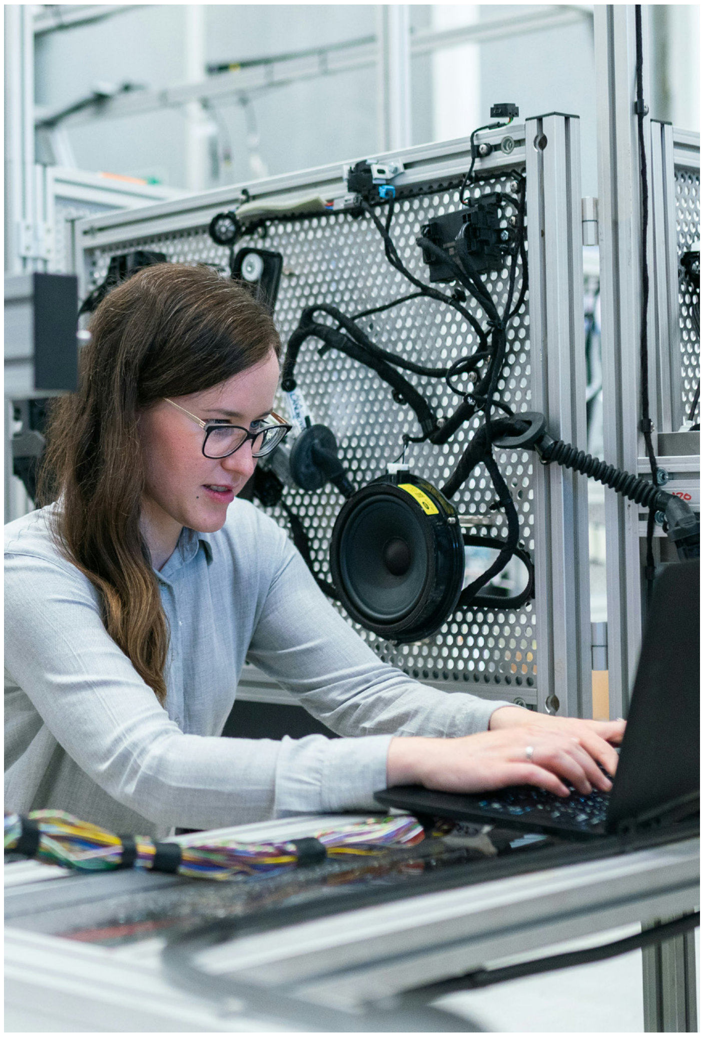
Google has shared guidance on how to craft an acceptable use policy for gen AI,[6] including recommendations for specifying scope, enforcement responsibilities, and delegating accountability for data governance and information security. Google has also published the Secure AI Framework (SAIF),[7] a conceptual framework to secure AI systems. Both of these may help inform your own policies.

What we've learned

DORA's research found AI adoption success in organizations who effectively communicate their vision and provide clear guidelines for AI usage. Based on this insight, we propose four strategies for increasing AI adoption among developers.

All of these strategies suggest that fostering widespread and effective AI adoption among developers requires a transparent, clear, and supportive approach. Organizations without a scaling strategy risk low adoption rates at best, and misuse of AI and AI tools at worst.

The strategies we recommend will help ensure that AI is used strategically and responsibly to enhance productivity, foster innovation, and drive organizational success.



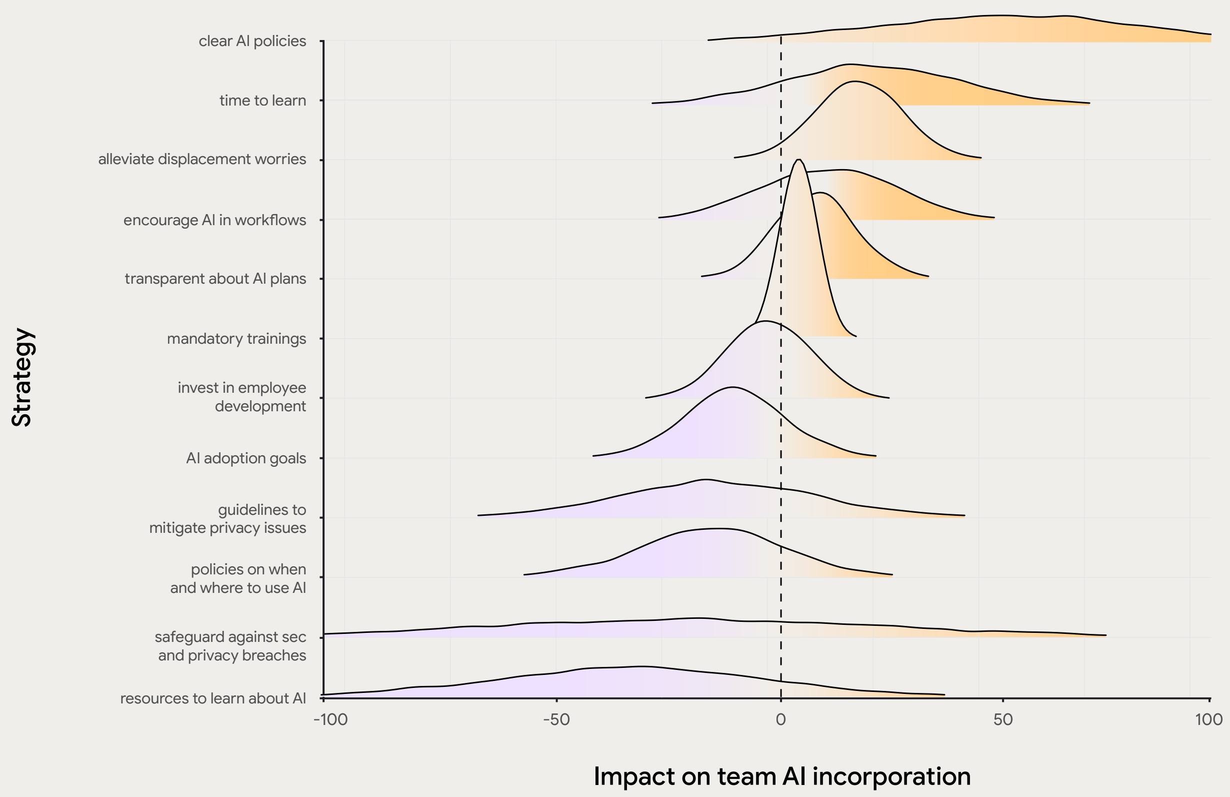


Figure 5: Impact of various strategies on AI incorporation

Figure 5 displays Bayesian posterior distributions, which represent our updated understanding of a value after considering new evidence. The height of each curve over a value indicates the relative plausibility of that value. Taller, narrower curves mean we're more certain about our estimates, while shorter, broader curves reflect greater uncertainty.

The value we're examining is how a specific strategy impacts a team's adoption of AI, measured by their response to the survey question: 'Over the last 3 months, approximately what percentage of your team's work is supported by AI?' Each curve represents 4000 simulated scenarios, comparing teams that haven't adopted the strategy at all with those that have thoroughly integrated it.

1. Accelerate State of DevOps Report 2024 - https://dora.dev/dora-report-2024
2. Accelerate State of DevOps Report 2023 - https://dora.dev/dora-report-2023
3. Build community structures to spread knowledge - https://dora.dev/guides/devops-culture-transform/#build-community-structures-to-spread-knowledge
4. How to craft an Acceptable Use Policy for gen AI (and look smart doing it) - https://cloud.google.com/transform/how-to-craft-an-acceptable-use-policy-for-gen-ai-and-look-smart-doing-it
5. Accelerate State of DevOps Report 2024 - https://dora.dev/dora-report-2024
6. How to craft an Acceptable Use Policy for gen AI (and look smart doing it) - https://cloud.google.com/transform/how-to-craft-an-acceptable-use-policy-for-gen-ai-and-look-smart-doing-it
7. Google's Secure AI Framework (SAIF) - https://safety.google/cybersecurity-advancements/saif/

Measuring success: Key metrics and feedback loops

Measurements help evaluate and guide improvement efforts. They provide important insights and signals that can be used to support decision making and prioritization. Measuring without acting on the signal is wasted effort. As your team adopts generative AI, you will need to assess how you're doing, prioritize improvements, and monitor progress.

Measurements should be taken at regular intervals and at different levels across an organization. In this chapter we will offer some suggestions for measuring at the team, service, and organizational levels.

Our suggestions should be seen as a starting point, and you should reevaluate which measures are useful for your team and organization.

Establishing some baseline measurements before you start or continue adopting generative AI is critical. Organizations and teams are complex and have a lot of emergent behaviors. We rarely have the ability to isolate change and fully quantify the impact of that change.

Capabilities and conditions for success

DORA's research has focused on identifying the capabilities and conditions that influence software delivery, operational, and organizational performance.

At a high-level, a climate for learning, fast flow, and fast feedback are the conditions that drive performance.

Capabilities
Climate for learning
Fast flow
Fast feedback
↓ Predict ↓
Performance
Software delivery Four keys metrics
Reliability Service Level Objectives (SLOs)
↓ Predicts ↓
Outcomes
Organizational performance
Well-being

Table 2: Summary view of the DORA Core model v2.0.0 - https://dora.dev/research/?view=summary

Each of these categories can be expanded to include specific capabilities, conditions, and metrics.

Capabilities

Climate for learning	Fast flow	Fast flow
Code maintainability	Continuous delivery	Continuous integration
Empowering teams to choose tools	Deployment automation	Resilience engineering
Documentation quality	Loosely coupled teams	Test automation
Generative culture	Version control	Monitoring and observability
	Database change management	Pervasive security
	Flexible infrastructure	Test data management
	Streamlining change approval	
	Working in small batches	

Performance

Software delivery	Reliability	
Measured by four key metrics:	Measured by Service Level Objectives (SLOs):	
Change lead time	Measurement coverage	Target optimization
Deployment frequency	Measurement focus	Target compliance
Change fail percentage		
Failed deployment recovery time		

Outcomes

Organizational performance	Well-being	
Commercial performance	Job satisfaction	Reduced burnout
Non-commercial performance	Productivity	Reduced rework

Table 3: Detail view of the DORA Core model v2.0.0 - https://dora.dev/research/?view=detail

Generative AI may be used directly in the pursuit of building up some of these capabilities, such as documentation quality[1] and test automation.[2] The use of gen AI may be a reflection of other capabilities such as empowering teams to choose tools.[3]

Similarly, some capabilities, such as generative culture,[4] may inform how gen AI tools are used in an organization.

We recommend using a mix of measurements to help guide your own adoption and improvement efforts.

Capabilities and conditions for success

Metrics drive conversations, support decisions, and help teams prioritize improvements. These measurements can be gathered in many ways including through conversations, surveys, and telemetry from systems. All are valid and each has varying levels of precision.

Our overall goal is to create a high-performing, sustainable technology-driven organization. Making progress toward that end is more important than improving the precision of measurements.

The metrics presented here are meant to assist you in understanding the impact of using generative AI throughout your software development and delivery lifecycle. Select the ones that seem best suited for where you are in your journey and complement these with some of your own metrics.

Code assistant metrics

Organizations leveraging AI coding assistants may want to measure the adoption using telemetry data directly from the tools. These can provide early indications of how adoption and use are progressing. Some measures that can be used include:

- **Licenses allocated:** what percentage of purchased licenses have been allocated to individuals?
- **Daily active users:** how many users are utilizing the gen AI tools on a daily basis?
- **Code suggestions:** how many code suggestions are being generated?
- **Chat exposures:** how many AI chat interactions are individuals having?
- **Code suggestions accepted:** how many code suggestions are being accepted?
- **Lines of code accepted:** how many lines of code are being accepted?

On their own, these metrics do not assess the impact of using coding assistants. Teams need a more comprehensive view of how the use of these tools is impacting teams, services, and the organization.

Fast-feedback metrics

Fast, quality feedback loops help build confidence and trust in new tools and processes. Feedback also helps individuals, teams, and organizations change tactics when they see that things aren't working as expected. Some measures that might help assess the speed and quality of feedback loops in your software delivery process include:

- **Tests on commit:** Do code commits result in a series of automated tests being run?
- **Daily tests:** Are automated tests executed at least daily?
- **Daily builds:** Is the application built automatically at least daily?
- **Test confidence:** When the automated tests pass, how confident is the team that the software is releasable?
- **Failures prevent progress:** Do automated test failures block a commit's progress through the pipeline?
- **Prioritize successful builds:** How long does it take between the build breaking and having it fixed, either with a check-in that fixes the problem, or by reverting the breaking change?

Team-level metrics

Teams can use these measures to identify patterns in their own work. Organizations can use these measures to identify and amplify successful patterns.

- **AI task reliance:** how much have you relied on AI for various tasks?
- **AI interactions:** how frequently have you interacted with AI?
- **AI productivity:** how has AI impacted your productivity at work?
- **AI ability to write code:** how has AI impacted your ability to write code?
- **AI trust:** how much did you trust the quality of the output from AI-generated code as part of your development work?
- **Organizational trust:** how much do you trust your organization to be transparent with how they plan on using AI?
- **Flow:** how often were you able to reach a high level of focus or achieve flow during development tasks?
- **Job satisfaction:** how do you feel about your job as a whole?

- **Productivity:** how productive do you feel at work?
- **Valuable work:** how much of your time at work was spent on work that you consider valuable?
- **Burnout:** how is your well-being at work?
- **Culture:** what is the culture like in your organization?

Service-level metrics

These metrics should be collected and evaluated at the level of an application or service. Cross-functional teams typically share responsibility for any single application or service, it is important to get representative insight from across the cross-functional team for each application or service being measured.

- **Code complexity:** how much has your team been hindered by code complexity if at all?
- **Code quality:** how satisfied is the team with the quality of code?
- **Code review time:** how long does it take to complete the code review process?
- **Documentation quality:** how is documentation quality for the service?
- **Technical debt:** how much is technical debt hindering productivity?
- **User centricity:** how is the team focused on the needs of users?
- **Product performance:** how is the product performing?
- **Software delivery performance:** what is the throughput and stability of software changes?

Organizational metrics

These metrics are at the organizational or business-unit level. These metrics help organizations understand the impact of the work of teams who are aligned to drive the mission of the organization.

- **Number of customers:** how many customers does the organization serve?

- **Market share:** what is the market share for the organization's primary products?
- **Performance:** how is the organization’s overall performance?
- **Profitability:** how is the organization’s overall profitability?
- **Mission:** how well is the organization achieving its goals?
- **Customer satisfaction:** how satisfied are your customers with the products and services your organization provides?
- **Operating efficiency:** how efficiently is the organization running?
- **Quality:** what is the quality of products or services provided by the organization?

Feedback loops

After collecting data and metrics, and analyzing them for insights, it’s time to take action to implement the lessons learned.

Fast, quality feedback loops help build confidence and trust in new tools and processes. Feedback also helps individuals, teams, and organizations change tactics when they see that things aren't working as expected.

Developer feedback loops

Learning how to use AI requires dedicated time for experimentation, practice, and integration with existing workflows. Feedback from these new tools is an important way to develop an understanding of how the tools can benefit existing practices, unlock new ways of working, and build trust in the tools.

Feedback begins with initial interactions with AI and flows through to monitoring and observability of the systems running in production serving end users.

Developers evaluate suggestions from coding assistants and decide whether to accept, refine, or reject the suggestion.

Fast, accurate suggestions help build confidence in the coding assistant which leads to additional usage.

Automated tests[5] help validate the code generated by developers and their AI-based coding assistants. These tests provide both feedback and guardrails as code is promoted through software delivery pipelines.

Software systems are complex, and an apparently simple, self-contained change to a single file can have unintended side effects on the overall system. When a large number of developers work on related systems, coordinating code updates is a hard problem, and changes from different developers can be incompatible.

The practice of continuous integration[6] (CI) helps address these problems. By creating rapid feedback loops and ensuring that developers work in small batches,[7] CI enables teams to produce high quality software, to reduce the cost of ongoing software development and maintenance, and to increase the productivity of the teams.

These and other practices and capabilities that enable fast feedback are important as teams begin using AI for coding assistance. We believe they'll become even more critical as teams enter the world of agentic workflows, where AI agents may generate code at a pace that current systems might not be equipped to handle. Continuous investment in faster feedback loops is crucial for ongoing success.

Team and organizational feedback loops

Teams can improve by gathering feedback from multiple sources that can help prioritize future actions. Here are four ways your team can capture feedback.

- **Regular surveys:** Use surveys to collect both quantitative and qualitative data on developer experiences with AI. Validated survey questions that you can reuse are available at https://dora.dev/research/ai/gen-ai-report-questions/

- **Retrospectives:** Incorporate AI adoption into team retrospectives. Discuss successes, challenges, and areas for improvement.

- **Communities of practice:** Facilitate knowledge sharing and the amplification of emerging practices through communities of practice.[8]

- **Data analysis:** Analyze the collected data to identify trends, correlations, and areas where interventions may be needed.

Use the insights gained from feedback loops to refine the AI adoption strategy, policies, training programs, and tool selection.

1. Documentation quality - https://dora.dev/capabilities/documentation-quality/
2. Test automation - https://dora.dev/capabilities/test-automation/
3. Empowering teams to choose tools - https://dora.dev/capabilities/teams-empowered-to-choose-tools/
4. Generative organizational culture - https://dora.dev/capabilities/generative-organizational-culture/
5. Test automation - https://dora.dev/capabilities/test-automation/
6. Continuous integration - https://dora.dev/capabilities/continuous-integration/
7. Working in small batches - https://dora.dev/capabilities/working-in-small-batches/
8. Build community structures to spread knowledge - https://dora.dev/guides/devops-culture-transform/#build-community-structures-to-spread-knowledge

A framework for gen AI adoption

This chapter outlines a framework for successfully adopting generative AI within an organization, focusing on both strategic leadership and practical implementation. We present a set of guiding principles centered around user-centricity, continuous improvement, transparency, responsible use, and iterative measurement, providing a solid foundation for integrating gen AI into workflows.

We provide actionable steps tailored for both organizational leaders and individual practitioners. The core concept being that all are responsible, at all levels, for the success of this adoption.

Guiding principles:

Successful AI adoption requires a shared understanding of core principles:

- **User-centeredness:** AI should serve the user's needs. For developers, this means that AI should be used for tools that enhance their workflow and reduce toil. For end-users, it means AI-powered features that improve the product experience.
- **Continuous improvement:** AI adoption is not a one-time event, but an ongoing process of experimentation, learning, and refinement. Embrace feedback loops and iterate based on data.
- **Transparency and communication:** Openly communicate the organization's AI strategy, goals, and policies. Address developer anxieties about job displacement directly. Foster a culture of psychological safety where developers feel comfortable experimenting and sharing their experiences.
- **Responsible use:** Establish clear organizational policies around AI usage.[1] Provide clarity regarding ethical considerations, data privacy, and security risks. This includes guidelines on when, where, and how to use AI, as well as mitigating and managing AI-related security risks.
- **Measurement and iteration:** Track key metrics to assess the impact of AI on teams, services, and the organization. Use this data to guide adjustments to the AI strategy and ensure it's delivering the desired results.

Actionable steps for leaders:

These seven steps can help leaders play a critical role in driving successful AI adoption.

1. **Create a vision and strategy:** Define a clear AI mission and adoption plan that aligns with overall organizational goals. Communicate this vision transparently to all employees, emphasizing the collaborative nature of human-AI partnerships.
2. **Empower teams:** Encourage experimentation with AI tools. Provide dedicated time during work hours for developers to learn, explore, and integrate AI into their workflows. This could take the form of hackathons, lunch-and-learns, or even a temporary reduction in productivity expectations. Support the formation of communities of practice where developers can share knowledge and best practices.
3. **Invest in training and resources:** Provide access to AI tools, training materials, and high-quality documentation. Ensure that internal

documentation is well-maintained and easily searchable, as AI can use this information to provide better context and support.

4. **Rethink performance metrics:** Reward outcomes and value delivered, not just hours worked. Acknowledge the labor involved in effectively working with AI, including prompt engineering and refining AI-generated output.

5. **Foster a climate for learning:** Encourage continuous learning and adaptation. Frame AI as an opportunity for developers to acquire new skills and expand their expertise.

6. **Manage expectations:** Be realistic about the timeline and potential challenges of AI adoption. Emphasize that this is an iterative process, and initial setbacks are to be expected.

7. **Establish clear policies:** Have clear guidelines about AI use, including when and where it should be used, and procedures for mitigating and managing security risks.

Actionable steps for practitioners:

Practitioners are using AI daily, and can improve how it benefits them by following these steps.

1. **Embrace experimentation:** Explore different AI tools and use cases within the established organizational guidelines. Focus on tasks where AI can provide immediate value, such as code generation, summarization, explaining unfamiliar code, code reviews, documentation, and writing tests.

2. **Develop "AI fluency":** Learn how to effectively interact with AI tools. This includes mastering prompt engineering techniques, understanding the limitations of AI, and integrating AI into your existing workflow.

3. **Prioritize quality and verification:** Treat AI-generated output as a starting point, not a finished product. Always review, test, and refine AI-generated code and documentation. Maintain a critical eye and rely on your expertise.

4. **Share knowledge and emerging practices:** Participate in communities of practice,[2] share your learnings with colleagues, and contribute to the organization's collective knowledge about AI.

5. **Focus on user impact:** Use AI to improve the user experience, whether that's through faster development cycles, higher-quality code, or more insightful analysis of user feedback.

6. **Advocate for responsible use:** Be mindful of ethical considerations, data privacy, and security risks. Follow organizational policies and guidelines, and speak up if you have concerns.

1. How to craft an Acceptable Use Policy for gen AI (and look smart doing it) - https://cloud.google.com/transform/how-to-craft-an-acceptable-use-policy-for-gen-ai-and-look-smart-doing-it
2. Build community structures to spread knowledge - https://dora.dev/guides/devops-culture-transform/#build-community-structures-to-spread-knowledge

Moving forward: Actionable takeaways and the future of AI in development

Generative AI is not just a trend; it's a fundamental shift in how software is built, tested, and maintained. This report has highlighted both the immense opportunities and the real challenges that organizations and developers face. To successfully navigate this transformation, consider these key takeaways:

For leaders

- **Prioritize transparency:** Clearly communicate your AI strategy, address job security concerns directly, and establish clear policies for responsible AI use.
- **Invest in people:** Provide dedicated time, training, and resources for developers to learn and experiment with AI. Foster a culture of learning and psychological safety.
- **Measure and iterate:** Track key metrics (code quality, developer satisfaction, delivery performance) and be prepared to adjust your approach based on data.

For practitioners

- **Embrace experimentation:** Don't be afraid to try new AI tools and explore different use cases within established guidelines.
- **Become AI-fluent:** Master prompt engineering, understand AI limitations, and integrate AI into your workflow.
- **Own the output:** Always review, test, and refine AI-generated code and documentation. Your expertise remains critical.

DORA's commitment

The journey of AI adoption is ongoing. The DORA team is committed to continuing our research and providing guidance to help organizations and developers thrive in this new era. We encourage you to:

- **Explore the resources:** Dive deeper into the 2024 Accelerate State of DevOps Report,[1] our ongoing research on enabling innovation with AI,[2] and other resources listed at the end of this report.
- **Join the community:** Connect with other professionals in the DORA Community[3] to share experiences, practices, and challenges.
- **Assess your performance:** Take the DORA Quick Check[4] to benchmark your current software delivery capabilities.

By embracing a data-driven, human-centered approach, we can collectively harness the power of AI to create a more productive, fulfilling, and innovative future for software development.

Next steps

Read the guide on "How to enable your software delivery teams to innovate with generative AI." https://dora.dev/guides/how-to-innovate-with-generative-ai/

Read more about how to transform your organization for perspective into the importance of ensuring teams have the tools and resources to do their job, and of making good use of their skills and abilities. https://dora.dev/guides/devops-culture-transform/

Explore the capabilities that enable a climate for learning, fast flow, and fast feedback. https://dora.dev/capabilities

Errata - Read and submit changes, corrections, and clarifications to this report. https://dora.dev/publications/errata

Check if this is the latest version of the Impact of generative AI in software development: https://dora.dev/vc/genai/?v=2025.2

Authors

Derek Debellis

Kevin M. Storer

Daniella Villalba

Nathen Harvey

Sarah D'Angelo

Adam Brown

This report was edited by Seth Rosenblatt

1. Accelerate State of DevOps Report 2024 - https://dora.dev/dora-report-2024
2. DORA Research: Artificial Intelligence - https://dora.dev/research/ai/
3. DORA Community - https://dora.community
4. DORA Quick Check - https://dora.dev/quickcheck/