Accelerate

# State of DevOps Report 2023

# Table of contents

All citations retrieved 27 September 2023

## Prelude

# Executive summary

For nearly a decade, the DORA research program has been investigating the capabilities and measures of high-performing technology-driven organizations. We've heard from more than 36,000 professionals from organizations of every size and across many different industries. Thank you for sharing your insights!

DORA tries to understand the relationship between ways of working (that is, capabilities) and outcomes: meaningful accomplishments that are relevant across an organization and relevant to the people in it. This research uses rigorous statistical evaluation and is platform-agnostic (see Methodology).

Our hope is that these insights will give leaders and practitioners a sense of where they can make an impact.

This year's research explored three key outcomes and the capabilities that contribute to achieving those outcomes:

- **Organizational performance**—The organization should produce not only revenue, but value for customers, as well as for the extended community.

- **Team performance**—The ability for an application or service team to create value, innovate, and collaborate.

- **Employee well-being**—The strategies an organization or team adopts should benefit the employees—reduce burnout, foster a satisfying job experience, and increase people's ability to produce valuable outputs (that is, productivity).

The research also explored means or performance measures that we often talk about like ends-in-themselves:

- **Software delivery performance**—Teams can safely, quickly, and efficiently change their technology systems.

- **Operational performance**—The service provides a reliable experience for its users.

# Key findings

**Establish a healthy culture**
Culture is foundational to building technical capabilities, igniting technical performance, reaching organizational performance goals, and helping employees be successful. Teams with generative cultures have 30% higher organizational performance.

**Build with users in mind**
A user focus can inform and drive improvements across all of the technical, process, and cultural capabilities we explore in our research. Teams can deploy as fast and successfully as they'd like, but without the user in mind, it might be for naught. Teams that focus on the user have 40% higher organizational performance.

**Unlock software delivery performance with faster code reviews**
Speeding up code reviews is one of the most effective paths to improving software delivery performance. Teams with faster code reviews have 50% higher software delivery performance.

**Amplify technical capabilities with quality documentation**
High-quality documentation amplifies the impact that technical capabilities have on organizational performance. Trunk-based development, for example, is estimated to have 12.8x more impact on organizational performance when high-quality documentation is in place relative to low-quality documentation.

**Increase infrastructure flexibility with cloud**
Cloud computing is beneficial because it creates a flexible infrastructure. Using a public cloud, for example, leads to a 22% increase in infrastructure flexibility relative to not using the cloud. This flexibility, in turn, leads to 30% higher organizational performance than inflexible infrastructures. To get the most value out of the cloud, the key is to take advantage of the differentiating characteristics and capabilities cloud has to offer, namely infrastructure flexibility.

**Balance delivery speed, operational performance, and user focus**
You need both strong software delivery performance and strong operational performance for organizational performance to see its fullest potential. Keeping these two balanced with a user focus yields the best organizational results while also improving employee well-being.

**Distribute work fairly**
People who identify as underrepresented and women or those who chose to self-describe their gender have higher levels of burnout. There are likely multiple systematic and environmental factors that cause this effect. Unsurprisingly, we find that respondents who take on more repetitive work are more likely to experience higher levels of burnout, and members of underrepresented groups are more likely to take on more repetitive work. Underrepresented respondents have 24% more burnout than those who are not underrepresented. Underrepresented respondents do 29% more repetitive work than those who are not underrepresented. Women or those who self-described their gender do 40% more repetitive work than men.

# Applying insights from DORA in your context

Teams that adopt a mindset and practice of continuous improvement are likely to see the most benefits.[1] DORA can help influence your own improvement initiatives.

To get the most out of this research, consider it in the context of your team and your users. For example, we stated earlier that teams with faster code reviews have 50% higher software delivery performance. However, your software delivery performance is unlikely to improve if your code reviews are already fast but speed is constrained elsewhere in the system. Contextualizing the research is possible when practitioners have conversations about how work is completed today. These conversations can lead to improved empathy, collaboration, and understanding of each participant's motivations.

Improvement work is never done. Find a bottleneck in your system, address it, and repeat the process. The most important comparisons are from looking at the same application over time, not by looking at other applications, organizations, or industries.

**Metrics and measurements**

Metrics and dashboards help teams monitor their progress and correct course.

Practitioners and leaders are striving for organizational performance, team performance, and well-being. But measurement is not the goal, just as delivering software is not the goal.

Fixating on performance metrics can lead to ineffective behaviors. Investing in capabilities and learning is a better way to enable success. Teams that learn the most improve the most.

## 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 site https://dora.community.

[1] 2022 Accelerate State of DevOps Report. https://dora.dev/research/2022/dora-report/2022-dora-accelerate-state-of-devops-report.pdf#page=7

# Concepts and measures

This section contains descriptions of the concepts DORA tries to measure.[1] These are the elements at the foundation of both this report and our models. These sections include the ingredients that we used to make this report. Hence, it is important that we, the authors, are clear about what these concepts are and that we are consistent in how we talk about them. The following tables are intended to provide clarity and ground all of us—readers and authors alike—in a shared terminology.

Because many of the concepts in this report are multifaceted, we often use multiple indicators to capture them. One way we evaluate our success of capturing these concepts is using exploratory factor analysis and confirmatory factor analysis. You can read more about those processes in Methodology. After evaluating our measurement methods, we scaled the scores from 0 to 10, with 0 representing the complete lack of presence of a concept and 10 representing the maximum presence of a concept. We believed this would standardize the way we talked about how these concepts function and help us compare data across the years.

Each concept we discuss will be accompanied with the following information:

- An icon to help convey meaning and hopefully make it easier to find when using this chapter as a reference.

- The average score for this concept in the sample (the mean).

- The boundaries of the interquartile range (IQR). By giving you the two numbers (25th and 75th percentiles) where the middle 50% of the data resides, these boundaries should help convey the spread of responses.

- The middle value in a set of data (the median). If it is dramatically different from the mean, it might indicate that the data is skewed.

- A description of the concept and how we measure it.

[1]Survey questions used in our analysis are published on https://dora.dev

# Key outcomes

Key outcomes are the goals that we believe people, teams, or organizations are striving to either reach (organizational performance, for example) or avoid (burnout, for example). As a result, we think the measures are important ways for people to evaluate themselves, their teams, and their organizations.

## Organizational performance

| Mean | IQR | Median |
|------|-----|--------|
| 6.3 | 5-8 | 6.3 |

High performing organizations have more customers, higher profits, and more relative market share for their primary product or service.

## Team performance

| Mean | IQR | Median |
|------|-----|--------|
| 7.6 | 6.6-9 | 8 |

High performing teams adapt to change, rely on each other, work efficiently, innovate, and collaborate.

## Software delivery performance

| Mean | IQR | Median |
|------|-----|--------|
| 6.3 | 5.1-7.8 | 6.4 |

The following four metrics measure the speed and stability of software delivery:

- Deployment frequency
- Lead time for changes
- Change failure rate
- Failed deployment recovery time

## Operational performance

| Mean | IQR | Median |
|------|-----|--------|
| 6.2 | 5-7.5 | 6.3 |

The extent to which a service is able to meet the expectations of its users, including measures like availability and performance.

## Reliability targets

| Mean | IQR | Median |
|------|-----|--------|
| 7 | 5-7.5 | 7.5 |

The extent to which a service meets its stated goals for measures like availability, performance, and correctness.

---

**Well-being** is a composite of burnout, productivity, and job satisfaction

## Burnout

| Mean | IQR | Median |
|------|-----|--------|
| 4.1 | 2-6 | 4 |

Not only the psychological and physical toll of work, but how one appraises the value and meaning of their work. Burnout causes cynicism. [2]

## Productivity

| Mean | IQR | Median |
|------|-----|--------|
| 7.5 | 6.7-8.8 | 7.9 |

A productive person does work that aligns with their skills, creates value, and lets them work efficiently.

## Job satisfaction

| Mean | IQR | Median |
|------|-----|--------|
| 6.08 | 5.7-7.1 | 7.1 |

A single-item question that asks the respondent to take everything into consideration and rate how they feel about their job as a whole. [3]

---

[2] Maslach C, Leiter MP. Understanding the burnout experience: recent research and its implications for psychiatry. World Psychiatry. 2016 Jun;15(2):103-11. doi: 10.1002/wps.20311. PMID: 27265691; PMCID: PMC4911781.

[3] Warr, P., Cook, J., & Wall, T. "Scales for the measurement of some work attitudes and aspects of psychological well-being." Journal of Occupational Psychology, 52(2), 1979. 129–148. https://doi.org/10.1111/j.2044-8325.1979.tb00448.x

# Processes & technical capabilities

These are either activities, practices, or states that might emerge in a team or organization. Put differently, these are things teams do or ways teams are.

## Artificial intelligence contribution

| Mean | IQR | Median |
|------|-----|--------|
| 3.3 | 0.3-6.3 | 2.4 |

The importance of the role of artificial intelligence in contributing to a variety of technical tasks.

## Loosely coupled architecture

| Mean | IQR | Median |
|------|-----|--------|
| 6.4 | 4.7-8.3 | 6.7 |

Software that can be written, tested, and deployed independently.

## Continuous integration

| Mean | IQR | Median |
|------|-----|--------|
| 6.9 | 5-8.9 | 7.8 |

The practice of automatically building and testing software changes.

## Continuous delivery

| Mean | IQR | Median |
|------|-----|--------|
| 7.0 | 5.7-8.7 | 7.3 |

The ability to get changes of all types—including new features, configuration changes, bug fixes, and experiments—into production, or into the hands of users, safely and quickly, and sustainably.[4]

## Code review speed

| Mean | IQR | Median |
|------|-----|--------|
| 6.5 | 6-8 | 6 |

A single item assessing the time it takes from the pull request to the code change review.

## Documentation

| Mean | IQR | Median |
|------|-----|--------|
| 5.8 | 3.8-7.9 | 6.25 |

The quality of written content that people in the organization create and use in their daily work.

## Reliability practices

| Mean | IQR | Median |
|------|-----|--------|
| 5.9 | 3.9-8.3 | 6.1 |

Activities and practices teams use to improve the operational performance of services.

## Trunk-based development

| Mean | IQR | Median |
|------|-----|--------|
| 5.6 | 3.9-7.8 | 5.6 |

The practice of making small, frequent changes that are regularly merged into the main code branch of the version control system.

## Flexible infrastructure

| Mean | IQR | Median |
|------|-----|--------|
| 6.6 | 5-8.3 | 7.3 |

Scalable infrastructure that is elastic, accessible, and measured.[5]

[4]"What is Continuous Delivery" https://continuousdelivery.com/

[5] National Institute of Standards and Technology (2018) NIST The NIST Definition of Cloud Computing. Available at https://csrc.nist.gov/pubs/sp/800/145/final

# Culture aspects

Defining culture isn't easy, but one might say that these are the prevailing norms (such as flexibility), the prevalent orientation (such as user centricity), and the ambience (such as organizational stability) of the workplace.

## Work distribution

| Mean | IQR | Median |
|---|---|---|
| 5.8 | 3.8-7.9 | 5.8 |

Formal processes to help employees distribute tasks equitably within a team.

## Flexibility

| Mean | IQR | Median |
|---|---|---|
| 7.7 | 6.6-8.9 | 8.3 |

How, where, and when a person works on tasks.[6]

## Job security

| Mean | IQR | Median |
|---|---|---|
| 5.9 | 3.3-8.3 | 6.7 |

A single-item measure that asks people how often they worry about their job security. Higher scores equal less worry.

## Organizational stability

| Mean | IQR | Median |
|---|---|---|
| 7.2 | 6.7-8.3 | 8.3 |

A single-item measure that asks how stable or unstable the work environment is for employees.

## Knowledge sharing

| Mean | IQR | Median |
|---|---|---|
| 6.4 | 5.0-8.3 | 6.7 |

How ideas and information spread across an organization. Team members answer questions once, and make the information available to others. People don't have to wait for answers.[7]

## User-centrism

| Mean | IQR | Median |
|---|---|---|
| 7.8 | 5.6-8.3 | 7.8 |

Understanding and incorporating users' needs and goals to make products and services better.[8]

## Westrum organizational culture

| Mean | IQR | Median |
|---|---|---|
| 7.3 | 6.1-8.6 | 7.8 |

How an organization tends to respond to problems and opportunities. There are three types of culture: generative, bureaucratic, and pathological.[9]

[6] Shifrin, Nicole V., and Jesse S. Michel. "Flexible work arrangements and employee health: A meta-analytic review." Work & Stress 36, no. 1, 2022. 60-85

[7] "2022 Developer Survey" https://survey.stackoverflow.co/2022#overview

[8] Kersten, Mik. Project to Product: How to survive and thrive in the age of digital disruption with the flow framework (IT Revolution, 2018), 54. https://itrevolution.com/product/project-to-product/

[9] Westrum R. "A typology of organisational cultures." BMJ Quality & Safety, 2004. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1765804/

# Chapter 1

# How do you compare?

> Goodhart's law:
> when a measure becomes
> a target it ceases to be
> a good measure.[1]

## Takeaways

The first step in improving performance is to set a baseline for an application's current software delivery performance, operational performance, and user-centricity. These measures help teams evaluate how they're doing and provide a good signal for how things are changing over time.

These measures, though, are not the means by which a team will improve. With this baseline, it is important to assess a team's strength across a wide range of people, processes, and technical capabilities to identify which might be holding back progress.[2] Next, teams need the time and space to align, experiment, and reassess. Repeating this process will help teams adopt a mindset and practice of continuous improvement.

[1] Strathern, Marilyn (1997). "'Improving ratings': audit in the British University system". European Review. John Wiley & Sons. 5 (3): 305–321. doi:10.1002/(SICI)1234-981X(199707)5:3<305::AID-EURO184>3.0.CO;2-4. S2CID 145644958.

[2] This report and the resources listed in "Capability catalog" (https://dora.dev/devops-capabilities/) can help.

[3] Forsgren, N., Storey, M-A.,et. al. "The SPACE of Developer Productivity: There's more to it than you think.," 2021. https://queue.acm.org/detail.cfm?id=3454124

Watch out for these and other pitfalls when using these comparisons:

- **Unlike comparisons.** Comparing applications based solely on these clusters is not likely to be useful. Doing so discards the context of each application in ways that might be detrimental to the goal of improving.

- **Setting metrics as a goal.** Ignoring Goodhart's law and making broad statements like "every application must demonstrate 'elite' performance by year's end" increases the likelihood that teams will try to game the metrics.

- **One metric to rule them all.** Attempting to measure complex systems with the "one metric that matters." Using a combination of metrics to drive deeper understanding.[3]

- **Narrowly scoped metrics.** People tend to measure what is easiest to measure, not what is most meaningful.

- **Using industry as a shield against improving.** For example, some teams in highly regulated industries might use regulations as a reason not to disrupt the status quo.

> For more details on our findings and advice for driving a mindset and practice of continuous improvement, see "How to transform" at:
> dora.dev/devops-capabilities/cultural/devops-culture-transform/

## Introduction

Each year we perform one or more cluster analyses to find common trends across applications. We recommend that you use these analyses to understand how you compare, but that you don't fixate on these comparisons. The best comparisons are those performed over time on the same applications rather than between different applications, which will always have different contexts.

Teams build software for users, and those users are the ultimate judges of the reliability and usefulness of the service. Teams that focus on the needs of users are better equipped to build the right thing. Combining user focus with software delivery performance and operations performance means those teams are also equipped to build the thing right.

> Teams that focus on the needs of users build the right thing AND build the thing right.

## Results

**Software delivery performance**

We use the following measures to assess software delivery performance:

- **Change lead time**—how long it takes a change to go from committed to deployed.

- **Deployment frequency**—how frequently changes are pushed to production.

- **Change failure rate**—how frequently a deployment introduces a failure that requires immediate intervention.

- **Failed deployment recovery time**—how long it takes to recover from a failed deployment.

A common approach to improving all four measures is reducing the batch size of changes for an application.[4] Smaller changes are easier to reason about and to move through the delivery process. Smaller changes are also easy to recover from if there's a failure. Teams should make each change as small as possible to make the delivery process fast and stable. Working in this way contributes to both change velocity and change stability.

[4] Oftentimes, a feature can be broken down into many changes that are independently delivered. Our measures of software delivery performance evaluate changes made to an application or service.

This year, we refined the measures of software delivery performance. Read more about those changes in "Refining how we measure software delivery performance" in the Appendix.

Here's a view into how this year's survey respondents are doing with software delivery performance:

| Performance level | Deployment frequency | Change lead time | Change failure rate | Failed deployment recovery time | % of respondents |
|---|---|---|---|---|---|
| Elite | On demand | Less than one day | 5% | Less than one hour | 18% |
| High | Between once per day and once per week | Between one day and one week | 10% | Less than one day | 31% |
| Medium | Between once per week and once per month | Between one week and one month | 15% | Between one day and one week | 33% |
| Low | Between once per week and once per month | Between one week and one month | 64% | Between one month and six months | 17% |

## Operational performance

We assessed operational performance by asking respondents how frequently their service does the following:

- Receives reports about end users being dissatisfied with the reliability of the system.

- Is unavailable, performs slower than expected, or performs incorrectly.

For an exploration of how operational performance predicts organizational performance, see Chapter 5 - Reliability unlocks performance.

Here's a view into how this year's survey respondents are doing with operational performance:



## User-centricity

A user-centric application or service is built with the end user in mind. Building a product like this requires a good sense of what users need and incorporating that into the product's roadmap. We assessed respondents' user-centricity by asking them the extent to which the following are true:

- Their team has a clear understanding of what users want to accomplish.

- Their team's success is evaluated according to the value they provide to their organization and to the users of the application.

- Specifications (for example, requirements planning) are continuously revisited and reprioritized according to user signals.

Here's a view into how this year's survey respondents are doing with user-centricity:
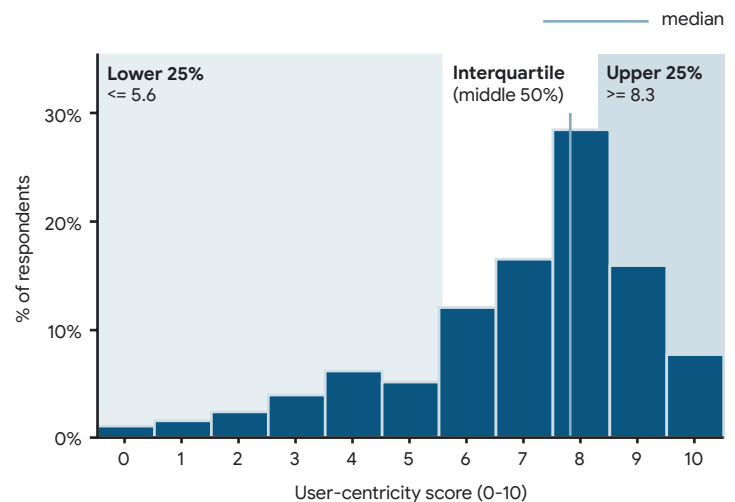
## Clustering team types

Comparing software delivery performance, operational performance, and user-centricity as a unit reveals four types of teams. These types of teams, like all of the measures that go into creating them, are at the level of an application or service.

We've named the team types User-centric, Feature-driven, Developing, and Balanced
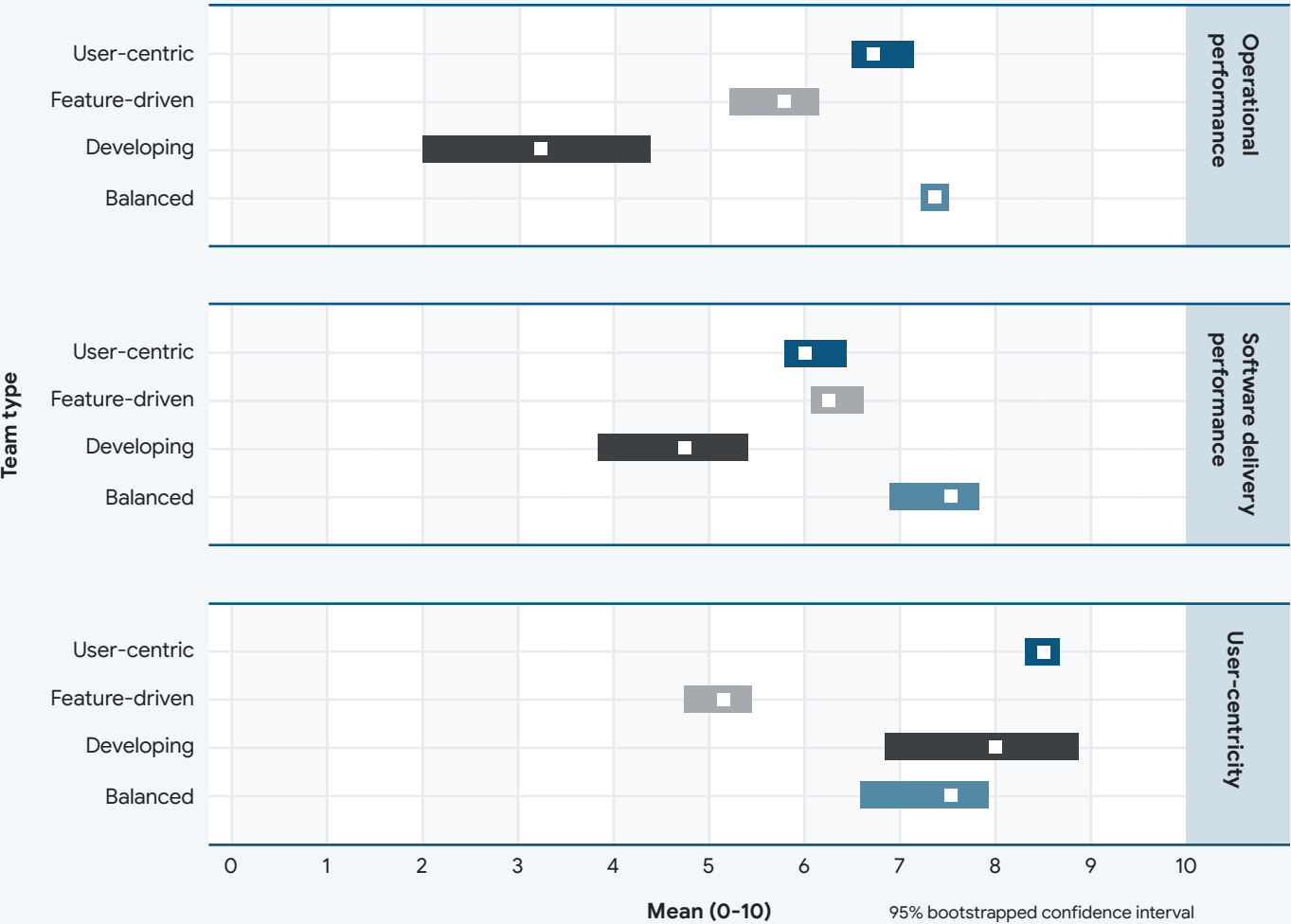
Think of the performance metrics we've been discussing as dials that an organization or team can adjust to change the organizational performance, team performance, and the well-being of the individuals on the team.

The graphs below show the performance outcomes predicted by each team type.
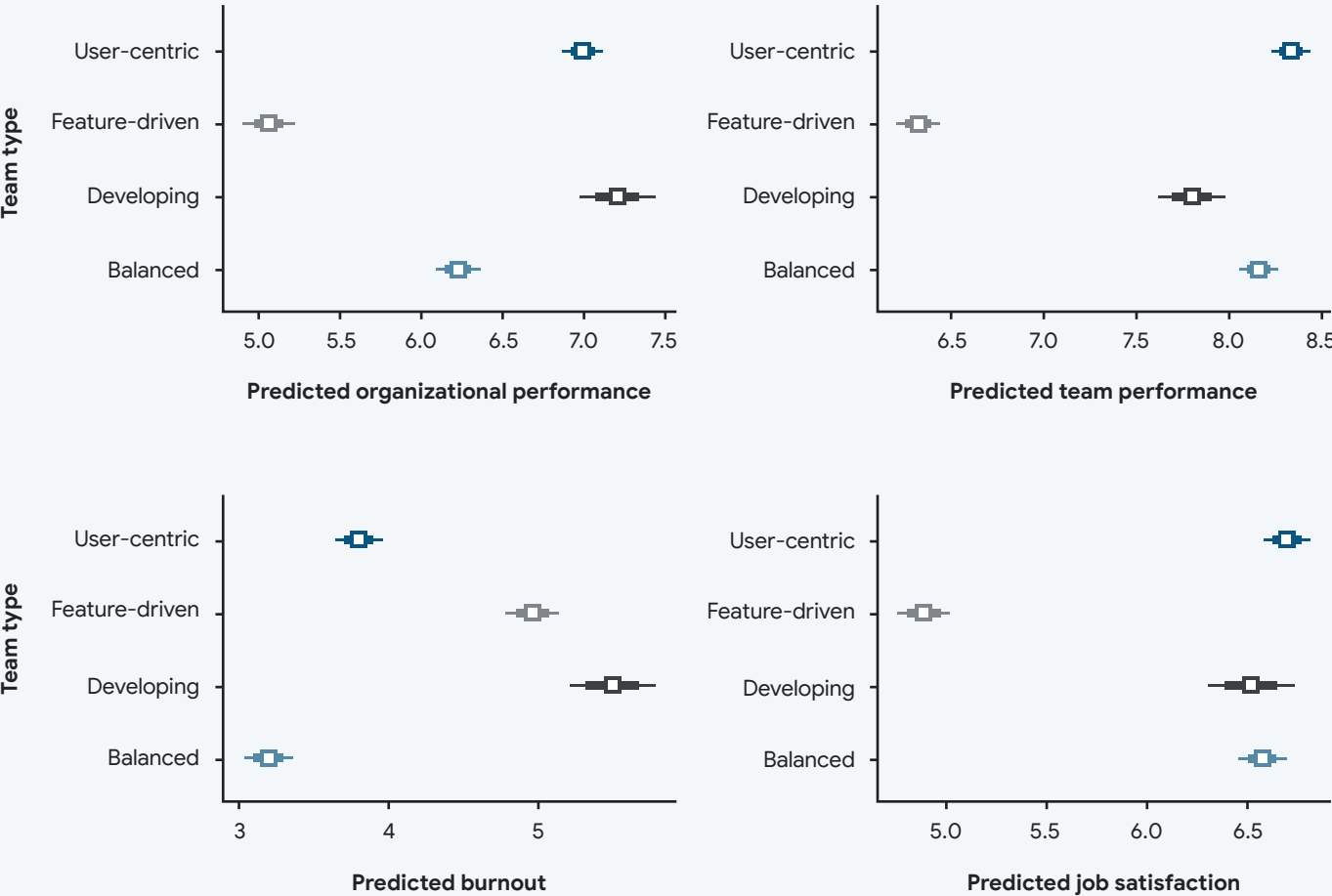


*Dot represents point estimate for team type mean. Thick interval where 66% of simulations fall. Skinny interval where 89% of simulations fall.

Each team type has unique characteristics, makes up a substantial proportion of our respondents, and has different outcomes. Your own team likely does not fit cleanly into only one, nor would we expect your team type to remain constant over time.
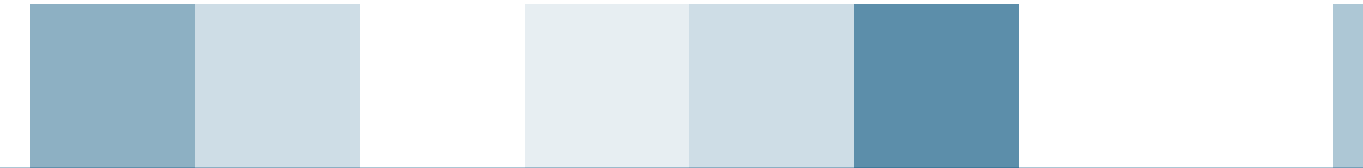
# What do these results mean?

**User-centric team type**

This team type shows the most focus on user needs. This focus, coupled with strong software delivery performance and strong operations performance, predicts the highest levels of organizational performance. However, it does show a bit more burnout than the balanced team. Improving software delivery performance and/or operations performance might be the best way for these teams to reduce burnout.

**Feature-driven team type**

This team type prioritizes shipping features. A relentless focus on shipping might distract the team from meeting users' needs, as demonstrated by lower user-centricity and operational performance numbers. These team types report some of the highest levels of burnout and the lowest levels of job satisfaction, of team performance, and of organizational performance. Employees value delivering value, not just features. Feature-driven teams might benefit from reflecting on the needs of their users as a way to get more value out of the features being shipped.

**Developing team type**

This team type achieves good organizational performance by focusing on the needs of the application's users. However, they might still be developing their product-market fit or their technical capabilities. These teams are more frequently found in smaller organizations. They demonstrate lower software delivery performance and operations performance, and teams working on these applications report higher levels of burnout than those in the balanced or user-centric team types. These teams might have heavyweight processes or toilsome tasks that could be automated as a way to improve their software delivery performance and operations performance.

**Balanced team type**

This team type demonstrates a balanced, sustainable approach. They're using technology in a sustainable way to achieve good organizational performance, good team performance, and good job satisfaction. These teams also report the lowest levels of burnout. These teams have adjusted their capabilities in a way that allows for good performance across all three measures. Increasing user-centricity might be their path to better organizational performance.

## Chapter 2
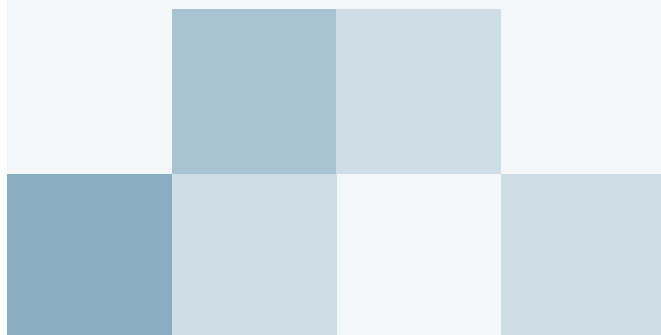
# Focusing on users predicts organizational performance

## Takeaways

Organizations can get caught up in the latest technology and management trends as they seek to improve developer productivity and organizational performance. Our research shows that a user-centric approach to building applications and services is one of the strongest predictors of overall organizational performance. To improve performance, develop a deep understanding of your users and iteratively adjust and incorporate their feedback.

**Teams with strong user-focus have**

# 40%

**higher organizational performance**

## Introduction

The DevOps movement started as a way to encourage better collaboration between development teams and operations teams in service of providing better user value. This alignment drove early successes and enabled DevOps ideas and capabilities to expand beyond those two departments. Today, high-performing technology-driven organizations recognize the importance of alignment across all teams to reach organizational goals.

We investigated three critical characteristics of being user focused:

- How well teams understand the needs of their users.

- How well aligned the team is toward meeting user needs.

- How user feedback is used when prioritizing work.

# Results

We found that a user-centric approach to software development leads to meaningful increases in performance. Organizations can experience a cascade of benefits when they put the user first. User feedback helps teams prioritize projects and helps them create products and services that meet user needs. This approach leads to a better user experience, increased user satisfaction, and increased revenue.

| Effect of user-centrism on … | | |
|---|---|---|
| Organizational performance | ⇈ | Substantial increase |
| Team performance | ⇈ | Substantial increase |
| Software delivery performance | ↑ | Minor increase |
| Operational performance | ⇈ | Substantial increase |
| Trunk-based development | ⇈ | Substantial increase |
| Reliability practices | ⇈ | Substantial increase |
| Continuous integration | ⇈ | Substantial increase |
| Continuous delivery | ⇈ | Substantial increase |
| Loosely coupled architecture | ⇈ | Substantial increase |
| Burnout | ↓ | Minor decrease* |
| Job satisfaction | ⇈ | Substantial increase |
| Productivity | ⇈ | Substantial increase |

*Reducing burnout is a good thing!

# What do these results mean?

Focusing on user needs is a strong predictor of overall organizational performance. Achieving strong user focus requires proper incentives, alignment, and ways of working. A user focus can inform and drive improvements across all the technical, process, and cultural capabilities we explore in our research.

Here's an exploration of how these results might affect various teams across your organization:

## » Product development and delivery teams

A focus on the user helps ensure that product development and delivery teams are building the right things for their users; hopefully while building them in a sustainable way. Balanced teams do just that. They demonstrate strength across delivery, operations, and organizational performance, with a strong focus on user needs. Members of these teams benefit from a clear understanding of user needs and the ability to adjust plans based on user feedback.

The results show that feature-driven teams failed to achieve top organizational performance. Such teams appear to be overly prioritizing delivery performance to the detriment of both organizational performance and the well-being of employees in the organization.

## » Operational teams

Teams that focus on operational performance might work hard to optimize system metrics like CPU use. But if they don't understand what a user expects of a service, they might still

get frequent user reports of slow performance. Site reliability engineering (SRE) practices, like identifying service level indicators that users care about and setting service level objectives that aim to keep a typical user happy, can help operational teams embody a more user-centered mindset.

**»  Platform engineering teams**
Platform engineering teams might adopt a "build it and they will come" approach to building out a platform. A more successful approach might be in treating developers as users of their platform. This shift in focus requires platform engineering teams to understand how developers work today to successfully identify and eliminate areas of friction. Teams can use the software delivery and operational performance measures as signals to monitor whether platform efforts are helping teams achieve better results.

**»  Leaders**
By creating incentive structures that reward teams for delivering value to users, leaders can help create an environment where a focus on the user thrives. Without these structures, teams might feel that they're merely measuring the number of features delivered or a reduction in service outages. DORA has investigated the role of transformational leadership[1] and has advice for leaders who are ready to improve this capability. For more information, see "DevOps Capabilities: Transformational Leadership" at https://dora.dev/devops-capabilities/cultural/transformational-leadership

**Focusing on the user leads to**

# 20%

**higher job satisfaction**

## Resources to get started

Developing more user-focused capabilities is an important driver of success. Our findings in 2023 reinforce our findings from 2018,[2] where we saw that lean product management capabilities predict software delivery performance and organizational performance.

Build your team's performance by adopting user-focus capabilities like customer feedback,[3] visibility of work in the value stream,[4] working in small batches,[5] and team experimentation.[6]

[1] 2017 State of DevOps Report. https://dora.dev/publications/pdf/state-of-devops-2017.pdf, 12-19

[2] 2018 Accelerate: State of DevOps Report: Strategies for a New Economy. https://dora.dev/publications/pdf/state-of-devops-2018.pdf, 49-51

[3] "Customer feedback." https://dora.dev/devops-capabilities/process/customer-feedback/

[4] "Visibility of work in the value stream." https://dora.dev/devops-capabilities/process/work-visibility-in-value-stream/

[5] "Working in small batches." https://dora.dev/devops-capabilities/process/working-in-small-batches/

[6] "Team experimentation." https://dora.dev/devops-capabilities/process/team-experimentation/

# Technical capabilities predict performance

## Takeaways

Investing resources and effort into continuous integration, a loosely coupled architecture, and increased code review speed will likely lead to many beneficial outcomes, such as improved organizational performance, improved team performance, improved software delivery performance, and improved operational performance. This happens without any detriment and often with some benefit to the well-being of individuals working on the application or service.

## Introduction

In the Executive Summary, we explained the technical capabilities that we studied and how they affected different performance and well-being measures. A central component of DORA has always been the exploration and quantification of the extent to which various processes and technical capabilities predict performance.

**This year, we investigated how the following technical capabilities predict performance:**

- Artificial intelligence

- Trunk-based development

- Loosely coupled architecture

- Continuous integration

- Rapid code review

**We looked at how they predicted these performance measures:**

- Team performance

- Organizational performance

- Software delivery performance

- Operational performance

**Additionally, we tested the connection between these capabilities and a number of indicators to determine how they affected the people doing the work:**

- Burnout

- Productivity

- Job satisfaction

# Results

The technical capabilities and processes that we studied have varied but overall positive effects on the key performance measures.

| Technical capabilities and processes | Effect on team performance | Effect on organizational performance | Effect on software delivery performance | Effect on operational performance |
|---|---|---|---|---|
| AI | ⊖ No effect demonstrated | ↑ Minor increase | ↓ Minor decrease | ↓↓ Substantial decrease |
| Continuous integration | ↑ Minor increase | ↑ Minor increase | ↑ Minor increase | ⊖ No effect demonstrated |
| Code review speed | ↑ Minor increase | ⊖ No effect demonstrated | ↑↑ Substantial increase | ↑↑ Substantial increase |
| Loosely coupled architecture | ↑↑ Substantial increase | ↑↑ Substantial increase | ↑ Minor increase | ↑↑ Substantial increase |
| Trunk-based development | ↑ Minor increase | ↑ Minor increase | ↑ Minor increase | ↓ Minor decrease |

Loosely coupled teams, or perhaps teams that have loosely coupled architecture, are able to make significant changes to their systems without involving other teams. This enables teams to move faster. When subject-matter experts are closer to the team, they can review code faster because they have a better understanding of the impact of the changes. A loosely coupled design enables the team to test, build, and deploy without other teams being a potential bottleneck.

Even with the smaller impact of our changes in a loosely coupled architecture, we need to ensure we're not introducing conflicts with the other developers on our team. Teams that work in small batches reduce the opportunities for conflict, ensuring that on each commit, the software is built, and automated tests are triggered, providing fast feedback to the developers.

Teams with shorter code review times have 50% better software delivery performance. Efficient code review processes lead to code improvements, knowledge transfer, shared code ownership, team ownership, and transparency.

Are code reviews your bottleneck? By evaluating your code review process and its effect on your lead time for changes, you can gain insights into opportunities for improvement. Consider the following questions:

- Are peer code reviews embedded in your process?

- How long is the duration between code completion and review?

- What is the average batch size of your code reviews?

- How many teams are involved in your reviews?

- How many different geographical locations are involved in your reviews?

- Does your team improve code quality automation based on code review suggestions?

Longer durations between code completion and review have been shown to negatively impact the effectiveness of the developer and the quality of software delivered. The involvement of multiple teams across geographical locations leads to longer duration, less engagement in the process, and increased costs.[1]

In 2022, we found that technical capabilities build on each other. Improving code review speed can contribute to improving several technical capabilities, including code maintainability, learning culture (knowledge transfer), and building a generative culture.[2]

Faster code reviews are one of the benefits of loosely coupled teams, leading to significant improvements to your software delivery performance and operational performance. There are several paths to improving the efficiency of your code reviews. When the code being reviewed only affects the scope of the team's architecture, the reviewer has a better understanding of the impact the code will have on the system. The smaller the code review, the easier it is for the reviewer to understand the implications of the change. Working in small batches improves the feedback cycle, efficiency, and focus for the team.[3] Pair programming is a practice that can reduce your code review time regardless of your current architecture and integration practices.[4]

Additionally, these capabilities and processes don't show a detrimental impact on the well-being of the individuals doing the work. In fact, most of these predict improvements to the individual's well-being.

---

[1] Investigating the effectiveness of peer code review in distributed software development based on objective and subjective data. https://jserd.springeropen.com/articles/10.1186/s40411-018-0058-0

[2] Expectations, Outcomes, and Challenges of Modern Code Review. https://dl.acm.org/doi/10.5555/2486788.2486882

[3] "Working in small batches." https://dora.dev/devops-capabilities/process/working-in-small-batches/

[4] "On Pair Programming." https://martinfowler.com/articles/on-pair-programming.html

| Technical capabilities and processes | Effect on burnout* | Effect on job satisfaction | Effect on productivity |
|---|---|---|---|
| AI | ↕ Minor decrease | ↑ Minor increase | ↑ Minor increase |
| Continuous integration | ⊖ No effect | ↑ Minor increase | ⊖ No effect |
| Code review speed | ⬇⬇ Substantial decrease | ↑ Minor increase | ↑ Minor increase |
| Loosely coupled architecture | ⬇⬇ Substantial decrease | ⬆⬆ Substantial increase | ⬆⬆ Substantial increase |
| Trunk-based development | ⬆⬆ Substantial increase | ⊖ No effect | ⊖ No effect |

*You might notice how the color scheme is flipped for burnout. This is because reducing burnout is a good thing!

We see that the use of loosely coupled architecture, continuous integrations, and efficient code reviews enable teams to improve their organizational outcomes while maintaining and sometimes improving their well-being.

When teams have the autonomy to improve and maintain a reliable system that delivers value to their users, they experience improved job satisfaction, team performance, and software delivery performance. Architecture plays a significant role in a team's ability to focus on the user and improve their software delivery. By starting small and focusing on the user, teams saw significant improvements across trunk-based development, loosely coupled architecture, continuous integration, continuous delivery, and SRE. To improve your technical capabilities, provide opportunities for team experimentation and continuous improvement.[5]

[5] "Team experimentation." https://dora.dev/devops-capabilities/process/team-experimentation

# Benefits of continuous delivery

**Author: Dave Farley**

The fundamental tenet of continuous delivery (CD) is to work so that our software is always in a releasable state. To achieve this, we need to work with high quality. That way, when we detect a problem, it is easy to fix, so that we can recover to releasability quickly and easily.

To keep our software in that golden, releasable state, we need to work to establish fast feedback and recover from failures very quickly.

As a reader of this year's report, I imagine that these ideas are sounding familiar. The metrics of **Stability** (change failure rate and failed deployment recovery time) are all about quality, and the metrics of **Throughput** (change lead time and deployment frequency) are all about feedback and ease of detection of any problem.

If you practice CD, then you will be scoring highly on **Stability & Throughput**. If you have high scores on **Stability & Throughput**, it is hard to imagine that you aren't also practicing CD in order to achieve those high scores.

This year's analysis includes a look at how capabilities drive performance by looking for mediators of each capability. CD—the ability to release changes of all kinds on demand quickly, safely, and sustainably—is a substantial mediator of many technical capabilities. In other words, these capabilities work because they create an environment that makes CD possible.

The practice of CD, in turn, provides the mechanism through which these capabilities can predict stronger software delivery performance.

Releasability is an important standard to meet in general for software development, which is why CD emphasizes it. Releasability matters because it is a subjective, but definite, and context-applicable, statement of quality. The degree of rigor that defines releasability might be different if we're working on safety-critical systems than if we're writing software for a cake shop. But in both cases, releasability defines that we've done everything that we deem necessary to say that this code is ready, good enough, and safe enough for release into the hands of users.

So optimizing to keep our changes releasable is also optimizing for a context-specific definition of minimum acceptable quality for our system.

Teams that prioritize getting and acting on high-quality, fast feedback have better software delivery performance.
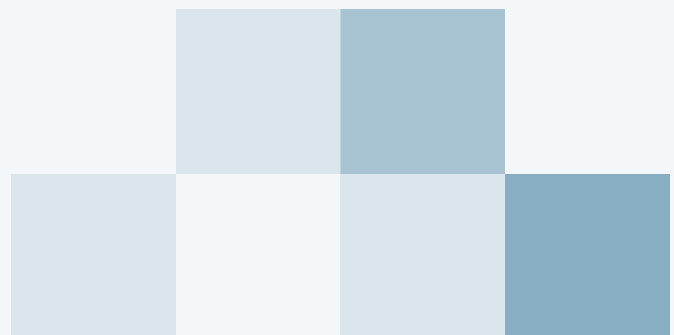
I am somewhat surprised that continuous integration (CI) and trunk-based development didn't have a bigger impact on software delivery performance. CI in particular seems pretty foundational to me, so this is somewhat challenging to my worldview, but teasing these things apart is complicated. For example, how can we achieve high scores on Throughput if our code doesn't integrate, and how can we be sure that our Stability is high if we haven't checked it? For me, CI is how we know these things, and so it's a key mediator of software delivery performance. Is this a problem of interpretation or something deeper and more important? Intriguing!

| Technical capabilities and processes | Effect on software delivery performance | Mediated through continuous delivery?* |
|---|---|---|
| AI | Minor decrease | No |
| Continuous integration | Minor increase | Yes, completely |
| Code review speed | Substantial increase | Yes, partially |
| Loosely coupled architecture | Minor increase | Yes, partially |
| Trunk-based development | Minor increase | Yes, completely |

*Mediation is a test that evaluates possible mechanisms or pathways underlying an effect. You can say, for example, "the data supports the hypothesis that the effect of trunk-based development on software delivery performance occurs through continuous deployment (the mediator)". Complete mediation is when the entire effect looks to be explained through the mediator. Partial mediation is when only some of the effect is explained through the mediator.

# Optimizing organizational processes and capabilities

We know that culture drives success. But what drives culture? This is an interesting question with everyone's favorite answer: It depends!

From a practitioner perspective, improving the way you work day-to-day has a positive impact on cultural elements such as sharing risk, increasing cooperation, and establishing psychological safety. For example, regularly integrating changes into the main branch of the version control system increases knowledge sharing and collaboration. Having security teams work alongside developers, collaborating on policy-as-code, increases trust between the teams and confidence in the changes being deployed.

From a leadership perspective, culture starts with awareness and education on the importance of culture. Transformational leadership[6] can help foster a blameless environment that encourages experimentation and learning, and gives trust and voice to the practitioners. Engineers are there to solve complex problems, not just respond to task requests. In order to do this, they need visibility into the business and the autonomy to take action. Ultimately, culture is downstream from leadership.

Ideally, the best results come from looking at culture from both a top-down and a bottom-up perspective.

[6] "Transformational leadership." https://dora.dev/devops-capabilities/cultural/transformational-leadership

# Artificial intelligence (AI)

Some analysts and technologists hypothesize that AI will make software teams more performant without negatively affecting professional well-being. So far our survey evidence doesn't support this. Our evidence suggests that AI slightly improves individual well-being measures (such as burnout and job satisfaction) but has a neutral or perhaps negative effect on group-level outcomes (such as team performance and software delivery performance).

We speculate that the early stage of AI-tool adoption among enterprises might help explain this mixed evidence. Likely, some large enterprises are testing different AI-powered tools on a trial basis before making a decision about whether to use them broadly. There is a lot of enthusiasm about the potential of AI development tools, as demonstrated by the majority of people incorporating at least some AI into the tasks we asked about. This is shown in the graph below. But we anticipate that it will take some time for AI-powered tools to come into widespread and coordinated use in the industry.

**Importance of AI**

**AI contribution to technical tasks**

For the primary application or service you work on, how important is the role of Artificial Intelligence (AI) in contributing to each of the following tasks today?



Legend:
Selected "Extremely Important"

Did NOT select "Not at all important"

*Interval provided for each graph represents 89% credibility interval. Provided to show inherent uncertainty in our estimates.
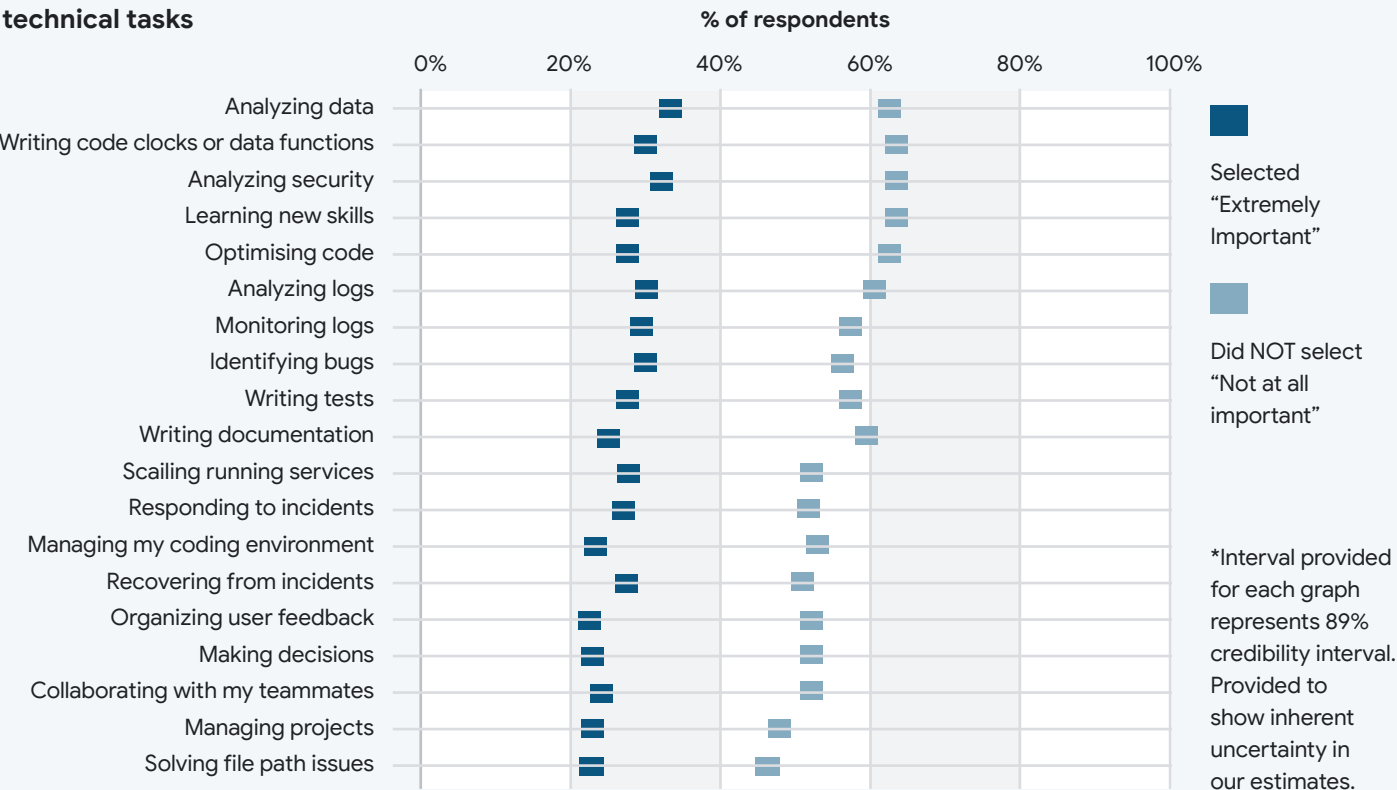
We are very interested in seeing how adoption grows over time and the impact that growth has on performance measures and outcomes that are important to organizations.

Chapter 4

# Documentation is foundational

## Takeaways

Quality documentation is foundational. It drives the successful implementation of technical capabilities and amplifies the impact those capabilities have on organizational performance. Documentation also has a positive impact on outcomes, such as team performance, productivity, and job satisfaction. However, increasing documentation quality doesn't lead to better well-being for everyone: as the quality of documentation increases, some respondents report increased levels of burnout.

## Introduction

This year we look deeper at internal documentation—the written knowledge that people in the organization use day-to-day. We investigate the impact of documentation on technical capabilities and on key outcomes.

To measure documentation quality, we measured the degree to which documentation is reliable, findable, updated, and relevant. We then calculate one score for the entire documentation experience. We're not evaluating documentation page-by-page, but as a whole.

# Results

**Documentation is foundational: it drives and amplifies technical capabilities**

As we found in 2021[1] and 2022[2], documentation quality continues to drive the successful implementation of every single technical capability we study.

As the following table shows, documentation quality also amplifies the impact of each technical capability on organizational performance, similar to what we saw in 2022[3].

| Technical capability | Amplification of impact on organizational performance* |
|---|---|
| Continuous integration | 2.4x* |
| Continuous delivery | 2.7x* |
| Trunk-based development | 12.8x* |
| Loosely coupled architecture | 1.2x* |
| Reliability practices | 1.4x* |
| Artificial intelligence contribution | 1.5x* |

*Calculated by: $\dfrac{\text{Impact of technical capability with high documentation quality}}{\text{Impact of technical capability with low documentation quality}}$

[1] Accelerate State of DevOps 2021. https://dora.dev/publications/pdf/state-of-devops-2021.pdf

[2] 2022 Accelerate State of DevOps Report. https://dora.dev/research/2022/dora-report/2022-dora-accelerate-state-of-devops-report.pdf

[3] 2022 State of DevOps Report data deep dive: Documentation is like sunshine. https://cloud.google.com/blog/products/devops-sre/deep-dive-into-2022-state-of-devops-report-on-documentation

**Documentation predicts productive and happy individuals and organizations**

In addition to improving technical capabilities, we found that quality documentation has a positive impact on an individual's well-being: it reduces burnout, increases job satisfaction, and increases productivity. We found that some of this impact is because quality documentation increases knowledge sharing.

This effect isn't a huge surprise. It's easier to get stuff done when you know how to do it, and work is less frustrating when knowledge is shared.

| Aspects of well-being | Effect of quality documentation | |
|---|---|---|
| Burnout | ⬇⬇ | Substantial decrease |
| Job satisfaction | ⬆⬆ | Substantial increase |
| Productivity | ⬆⬆ | Substantial increase |

Quality documentation also drives key outcomes, affecting team performance, organizational performance, and operational performance.

| Key outcomes | Effect of quality documentation | |
|---|---|---|
| Team performance | ⬆⬆ | Substantial increase |
| Organizational performance | ⬆⬆ | Substantial increase |
| Software delivery performance | ⊖ | No effect* |
| Operational performance | ⬆⬆ | Substantial increase |

* We continue to be surprised to find no effect of quality documentation on software delivery performance. This is the second year we see this behavior.

What's behind this positive effect on the three key outcomes? As a reader, using clear documentation is beneficial. The writing process might be a factor as well. Creating high-quality documentation requires teams to decide on processes in the first place. Documentation can force teams across an organization to explicitly discuss and get aligned on what to do and how to do it.

Quality documentation also acts as a repository for team knowledge, even as people come and go. It helps knowledge scale, both throughout the organization and through time.

**Is documentation tied to decreased well-being for some people?**
We noticed an unexpected trend when we looked at respondents who identify as underrepresented. For this group, documentation quality is tied to an increase in burnout.

For this finding, we also looked at gender, and were surprised to find no effect. Respondents who identified as male, female, or self-described their gender all saw a significant reduction in burnout with high-quality documentation. However, people who identified as underrepresented, regardless of gender identity, noted a higher rate of burnout in the presence of quality documentation.

The following graph shows simulated predictions based on our data. In the lower set, we see burnout decrease for the majority of respondents as documentation quality increases. However, in the higher set, we see burnout significantly increase for individuals who identify as underrepresented.

This graph shows 1,000 simulated lines for each group. More densely packed lines mean that the slope is more likely given our data.
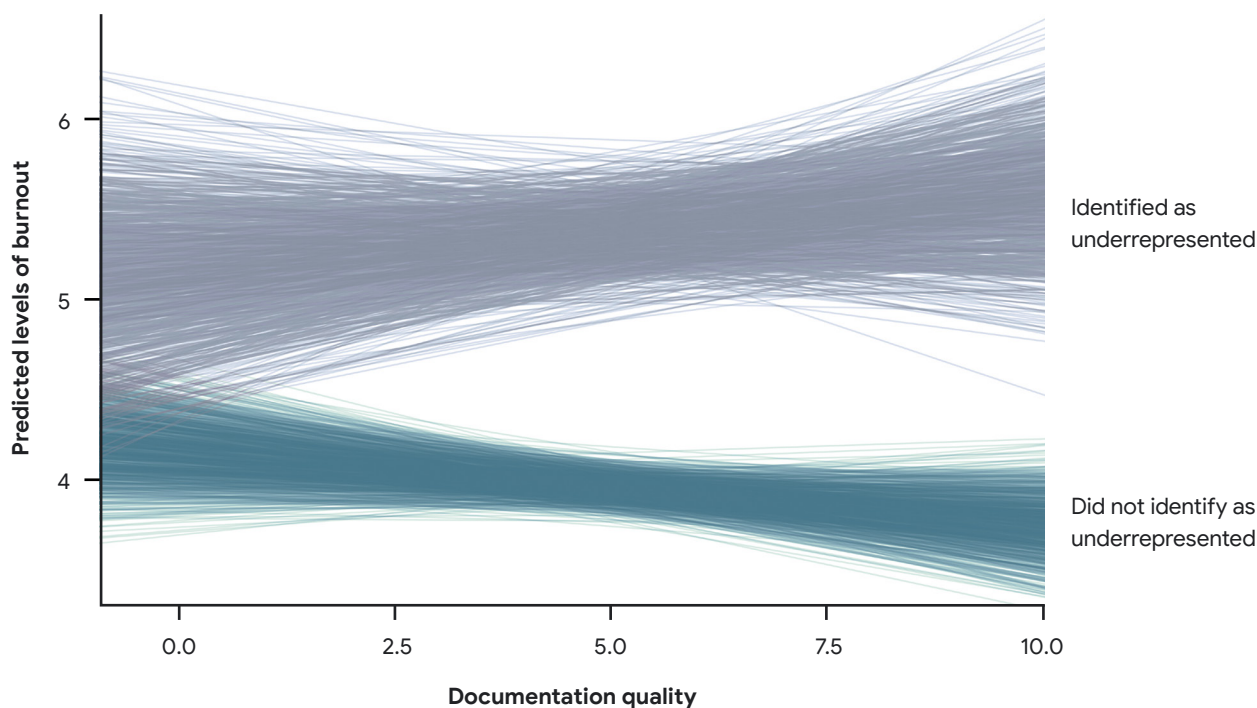
This finding is similar for documentation quality, generative culture, and team stability: as these attributes increase, burnout also increases for people who identify as underrepresented. For documentation, what's going on?

It takes work to create and maintain high-quality documentation. This is technical work, with significant impact on technical capabilities, team productivity, and organizational performance. It's also work that might not be consistently recognized for the importance and impact that it has. Are people who identify as underrepresented doing a disproportionate amount of this work, and if so, does this work help explain the effect on burnout?

Could the reliance on using documentation be problematic? With increased documentation quality, does knowledge sharing not increase for some respondents? Or, if it does increase, is it not enough to counteract other aspects that lead to burnout for this group?

It's possible that something else entirely is at play that drives quality documentation but also creates or maintains burnout for respondents who identify as underrepresented. More research is needed.

It seems that who you are on the team matters. Aspects of the workplace like quality documentation have significant benefits to the team and to the overall organization. But they might also be tied to negative outcomes for some individuals. We explore this more in Chapter 8 - How, when, and why who you are matters.

---

⁴ Accelerate State of DevOps 2021, 22. https://dora.dev/publications/pdf/state-of-devops-2021.pdf#page=22

## Resources to get started

See the 2021 report for practices that drive quality documentation.[4] This year, we also found that work distribution, including formal processes to distribute documentation work, significantly increase the quality of documentation.

**Lots of resources and training exist for technical writing. You can learn more from these resources:**

- Society for Technical Communications (stc.org)

- Technical Writing Courses for Engineers (developers.google.com/tech-writing)

- Write the docs (writethedocs.org)

Chapter 5
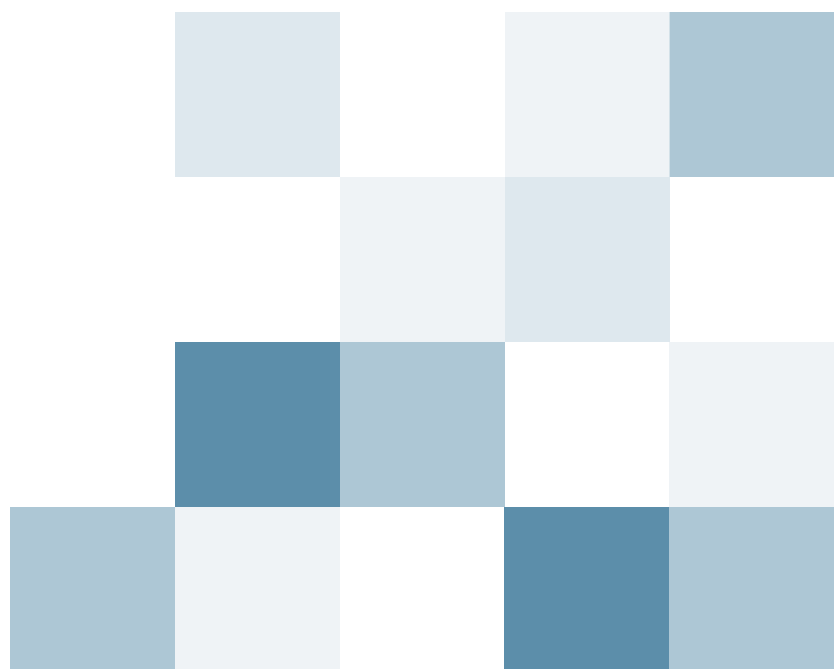
# Reliability unlocks performance

## Takeaways

Strong reliability practices predict better operational performance, team performance, and organizational performance. The data shows that the effects of improving these practices follow a nonlinear path— that is, there might be times when performance improvements seem to stall as organizations build stronger capabilities. However, over time, staying committed to these practices still predicts good outcomes.

## Introduction

Reliability is a widely used term in the IT operations space. We define reliability as the extent to which a service meets its stated goals for measures like availability, performance, and correctness. A common approach to achieve reliability outcomes is SRE, which originated at Google (https://sre.google) and is now practiced in many organizations. SRE prioritizes empirical learning, cross-functional collaboration, extensive reliance on automation, and the use of measurement techniques, including service level objectives (SLOs).

Many organizations use reliability practices without referring to them as SRE; alternative terms include Production Engineering, Platform Teams, Infrastructure Teams, TechOps, and others. In order to assess the extent of these practices as objectively as possible, our survey uses neutral, descriptive language in the survey text.

We also collect data on the outcomes of reliability engineering—the extent to which teams are able to achieve their reliability targets. Both reliability practices and reliability outcomes (which we refer to as **operational performance**) are reflected in our predictive model alongside other capabilities.

**Reliability practices**
We asked respondents to think about reliability by having them think through three essential aspects of their operations. First, do they have mitigation plans for their dependencies? Second, do they regularly test their disaster recovery plans through either simulated disruptions, practical failovers, or table-top exercises? Finally, when they miss their reliability targets, do they perform improvement work or otherwise reprioritize and adjust their work?

We think these measures encapsulate the spirit of a team that follows established SRE principles such as "embracing risk" and "measuring user happiness." Such a team sets a reasonable goal that aligns with user happiness. They then perform tests to ensure they're able to meet that goal, but they change plans if they're having trouble. We use this as a proxy for a team that's successfully "doing SRE" without tying assessments of teams to particular SRE implementations.

## Results

**Confirming the J-curve of reliability practices**

Since 2018, DORA has theorized that there's a nonlinear relationship (Figure 1) between operational performance and practices like automation. As we've deepened our explorations into reliability practices, we've seen evidence of this pattern in the survey data.

In 2022 we measured this directly. We surveyed teams and observed that the relationship between reliability practices and reliability outcomes did indeed follow this type of non-linear curve (Figure 2). This suggested that teams saw significant reliability gains only after they adopted many reliability practices.[1] But seeing the data in this way didn't feel like we were seeing the full picture. The 2022 curve made it feel like SRE is only for experts or otherwise not worth investing in, which conflicts with the experience of many SRE teams. We needed more data.

**Figure 1: 2018 hypothetical J-curve**



Automation helps low performers progress to medium performers.

Relentless improvement work leads to excellence and high performance! High and elite performers leverage expertise and learn from their environments to see jump in productivity.

Automation increases test requirements, which are dealt with manually. A mountain of technical debt blocks progress.

Teams begin transformation and identify quick wins.

Technical debt and increased complexity cause additional manual controls and layers of process around changes, slowing work.

[1] 2022 Accelerate State of DevOps Report. 27-28. https://dora.dev/research/2022/dora-report/2022-dora-accelerate-state-of-devops-report.pdf#page=27

**Figure 2: 2022 curve**



**Figure 3: 2023 curve**
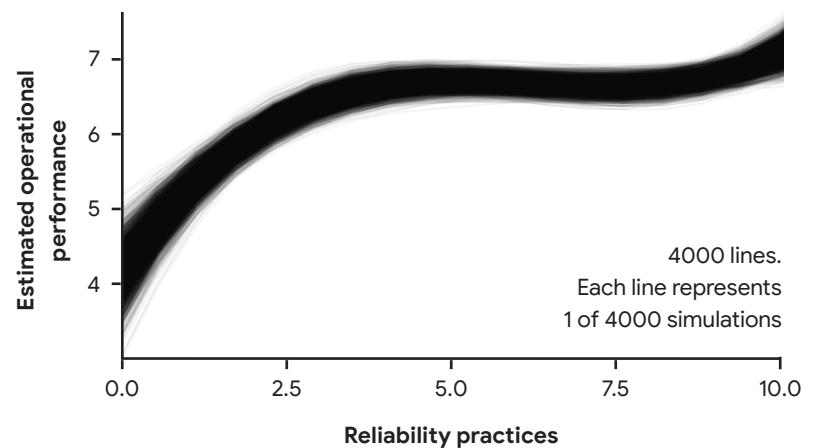


In 2023, we were able to ask more questions, which helped better define a curve that more closely matches our lived experiences. The new curve is closer to the hypothetical J-curve of transformation described in the 2018 report (see the Methodology section for more on how we perform our analysis). This suggests that there are indeed early wins in adopting reliability practices, followed by a lull as complexity introduces new challenges, and then finally another uptick in operational performance. The results reinforce what we've seen with many teams.

This curve matters for a few reasons:

- It helps companies rationalize and fund initial SRE adoption, even if they're not looking for extreme levels of reliability or don't expect to significantly invest in SRE. Adopting even small levels of reliability practices can result in operational performance improvements, which has further beneficial effects on team performance and organizational performance.

- It prepares companies who *are* looking to heavily invest in reliability to stick it out through the lull. It can be tempting to expect linear positive results from long-term investment in SRE, but the data tells us that isn't the case. When teams know about the nonlinearity of this curve ahead of time, they can make a decision about whether to make this investment, and they can plan ahead to ensure they don't abandon it before realizing the full benefits.

- Changes like this might require cultural transformation.[2] We have found success comes from a combination of bottom-up and top-down change. Teams can adopt reliability practices and reap the immediate rewards, then those benefits can be shown off to other teams, reinforced and incentivized by leadership. These incentives and structured programs can be designed with the J-curve in mind.

---

[2] "How to transform" - https://dora.dev/devops-capabilities/cultural/devops-culture-transform/

## Reliability practices and well-being

Traditional operations practices are highly reactive and often more concerned with the health of the technical system than with the happiness of its users. On-call alerts for things that don't impact users' experiences, repetitive manual tasks, fear of making mistakes, and similar experiences lead to burnout and poor well-being for individuals on the team.

We see the opposite in teams that leverage reliability practices. Teams report higher productivity and job satisfaction and lower levels of burnout than their counterparts who are not using these practices. We suspect that these improvements in well-being are driven by some published SRE practices:

- Reducing toil [3]

- Blameless postmortems [4]

- Team autonomy [5]

- Sublinear scaling of teams [6]

[3] Beyer, Betsy, et al. Site Reliability Engineering: How Google Runs Production Systems (O'Reilly, 2016), 49–54. https://sre.google/sre-book/eliminating-toil/

[4] Ibid, https://sre.google/sre-book/postmortem-culture/

[5] Beyer, Betsy, et al. The Site Reliability Workbook (O'Reilly, 2018), https://sre.google/workbook/team-lifecycles/

[6] Brookbank, James, and McGhee, Steve. Enterprise Roadmap to SRE (O'Reilly, 2022), 11. https://sre.google/resources/practices-and-processes/enterprise-roadmap-to-sre/

## Operational performance

We also asked respondents to describe the operational performance of their service. First, we asked how frequently they hear directly from their users about dissatisfaction in the reliability of their service. Next, we asked them how often their service is unavailable, slow, or otherwise operating incorrectly.

**»  Reliability practices amplify team and organizational performance, through operational performance**

By adopting reliability practices, teams improve their operational performance. If an organization is able to operate its production fleet effectively, we found that this *amplifies* other outcomes. If the outcomes are high, reliability practices will make them higher. If outcomes are low, reliability practices will not help, they will just stay that way.

Reliable systems still need to have the right software capabilities for your customers, delivered effectively. This makes sense because SRE was never intended to operate in a vacuum. Meeting reliability targets is a key metric of success for SRE teams, and this is reflected in operational performance. Although there are likely other benefits to the use of reliability practices, the data suggests that the most critical one is the impact on operational performance. Furthermore, increased operational performance has benefits beyond service health; in fact, we see evidence that the use of reliability practices predicts greater well-being for practitioners.

**»  Operational performance affects well-being**
A common industry perception is that highly reliable services have a negative impact on the well-being

of the service's operators, for example through on-call activities or emergency maintenance outside of work hours. However, we found that high operational performance actually results in lower burnout, better productivity, and higher job satisfaction. This aligns with the SRE principle of reducing toil;[7] automating the manual parts of operations is satisfying for individuals and also results in reduced ongoing burden for the team.

## ≫ Organizational performance and team performance amplify operational performance

We found that operational performance has a substantial positive impact on both team performance and on organizational performance. This shouldn't be a surprise to followers of the DevOps movement. Being able to operate the machine effectively allows teams to achieve more, which allows organizations to thrive.

## ≫ Operational performance amplifies software delivery performance

While software delivery performance can improve both team performance and organizational performance, they are both significantly enhanced by operational performance. Moreover, high-performing software delivery teams won't achieve very high team performance and organizational performance without also achieving high operational performance. Both are needed. In fact, teams that improve their software delivery performance without corresponding levels of operational performance end up having worse organizational outcomes. So, if you can quickly write stunning software but it fails to run in production in a way that meets its audience's expectations, there won't be any reward from the marketplace.

# What's missing, what's next?

We believe there are more measurements that can help us understand these interactions. For example, a common question this year has been how cost management plays into these capabilities and outcomes. Some organizations are more cost-sensitive than others, and this has implications for how the organization makes plans and decisions. Similarly, we theorize that reliability practices might emerge from highly collaborative cultures,[8] even without being explicitly sought after or planned for. We want to get a better understanding of how teams evolve their existing IT operations practices, and how that evolution affects system reliability, team performance, and well-being.

Mostly, we want to hear from you. Come join us and other practitioners at DORA.community.[9] SRE is still a new field. Its impact is different in every organization that adopts reliability practices, or in organizations that realize that they have been doing SRE this whole time. These changes are slow, and we want to make consistent measurements to show progress over time. As a community, we can share what works, elevating each other along the way.

---

[7] Beyer, Betsy, et al. Site Reliability Engineering: How Google Runs Production Systems (O'Reilly, 2016), 49–54. https://sre.google/sre-book/eliminating-toil/

[8] Brookbank, James, and McGhee, Steve.  Enterprise Roadmap to SRE (O'Reilly, 2022), 5. https://sre.google/resources/practices-and-processes/enterprise-roadmap-to-sre/

[9] DORA Community, https://dora.community/

# How Google does SRE

Within Google, SRE has two decades of evolution within a growing and well-funded organization. The impetus for SRE was a need to enable hypergrowth of Google Search and Ads without breaking the bank. Early on, these products' realtime nature revealed a need for high reliability—a transient error in search or ads is an immediate lost customer, and there isn't a chance to retry. A dynamic ad has to be calculated in milliseconds; a slow search degrades the entire promise of Google's brand.

At the same time, SRE was being developed in the cocoon of a new type of company: a bottom-up, engineering-led company that chose to build over buy. Google leveraged a staff of site reliability engineers (SREs) who were strong in academic computer science subjects like distributed systems and compiler design. This cultural DNA provided a rich environment for SRE to emerge and thrive. The SRE team was entrusted not only with the keys to production, but with finding new and novel ways to scale systems.

But how did SRE scale up over time? In a word: sublinearly. That is, Google couldn't double the number of SREs employed every time Google got two times bigger. Given the pace at which early Google products were scaling up to meet global demand (and with the introduction of new products like Gmail, Google Maps, Android, YouTube, and Google Cloud), it wasn't possible to scale these new, scarce SREs at the same rate at which the customer base grew.

A saying was born:

- SRE shall not scale linearly with the number of users.

- SRE shall not scale linearly with the number of servers.

- SRE shall not scale linearly with the number of clusters.

- SRE shall not scale linearly with the number of services.

A management structure developed to allow this constrained growth model to be maintained. SREs aligned and cooperated with each other while keeping in close step with the product development teams they worked with. SREs reported up through their own chain of management, with Ben Treynor-Sloss [10] at the top. These teams segmented themselves into product areas (PAs) that exactly aligned with product development ("dev") PAs. Teams of SREs worked with their dev teams to decide how best to use the SREs at their disposal.

---

[10] Benjamin Treynor Sloss, Vice President of Engineering. Site Reliability Engineering: How Google Runs Production Systems (O'Reilly, 2016). https://sre.google/sre-book/part-I-introduction/, https://sre.google/sre-book/introduction/#id-2opuzSjFr

Dev teams could fund new SREs directly. Not all dev teams created user-facing products; many were shared infrastructure teams like Bigtable [11] (structured data storage), Borg [12] (compute scheduling), and Colossus [13] (distributed storage). These shared infrastructure services then allowed customer-facing teams to scale without their own dedicated SRE team.

By keeping these teams in their own organization, teams were able to maintain a consistent hiring and promotion process. SRE teams tended to be dwarfed by their dev teams by a factor of ten or more, so it was important to ensure the SRE teams had autonomy and weren't pulled in any directions that went counter to SRE principles.

SREs developed their own internal products, complete with internal product managers. The customers for SRE products were other teams who wanted better ways to run production. SRE teams developed products around incident response, monitoring and observability, release management, capacity planning, and troubleshooting.

The process by which SRE teams grew always took team health and sustainability into account. For example, a new on-call SRE team had a minimum size of 12: two sites of 6 members. This allowed cross-timezone coverage, and having a sufficient number of people in a team allowed a good work-life balance without sacrificing anybody to burnout.

SRE continues to adapt today. Not every team has precisely followed the models described above, and some have chosen alternative paths. Large organizations like SRE within Google are also affected by market conditions just like any other large group, so flexibility is important. But above all, SRE teams stick to their principles: embracing risk, measuring service levels, eliminating toil, embracing automation, and striving for simplicity.

[11] Chang, Fay, et al. Bigtable: A Distributed Storage System for Structured Data, 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI), {USENIX} (2006), pp. 205-218, https://research.google/pubs/pub27898/

[12] Verma, Abhishek, et al. Large-scale cluster management at Google with Borg, Proceedings of the European Conference on Computer Systems (EuroSys), ACM, Bordeaux, France (2015), https://research.google/pubs/pub43438/

[13] Hildebrand, Dean, et al. Colossus under the hood: a peek into Google's scalable storage system. April 19, 2021 - https://cloud.google.com/blog/products/storage-data-transfer/a-peek-behind-colossus-googles-file-system

# Flexible infrastructure is key to success

## Takeaways

Flexible infrastructure is a predictor of team performance, organizational performance, operational performance, and software delivery performance. Cloud computing is a core enabler of flexible infrastructure, but this benefit isn't automatically realized: our data shows that *how* you use the cloud is the important part.

**Flexible infrastructures predict**

# 30%

**higher organizational performance than inflexible infrastructures**
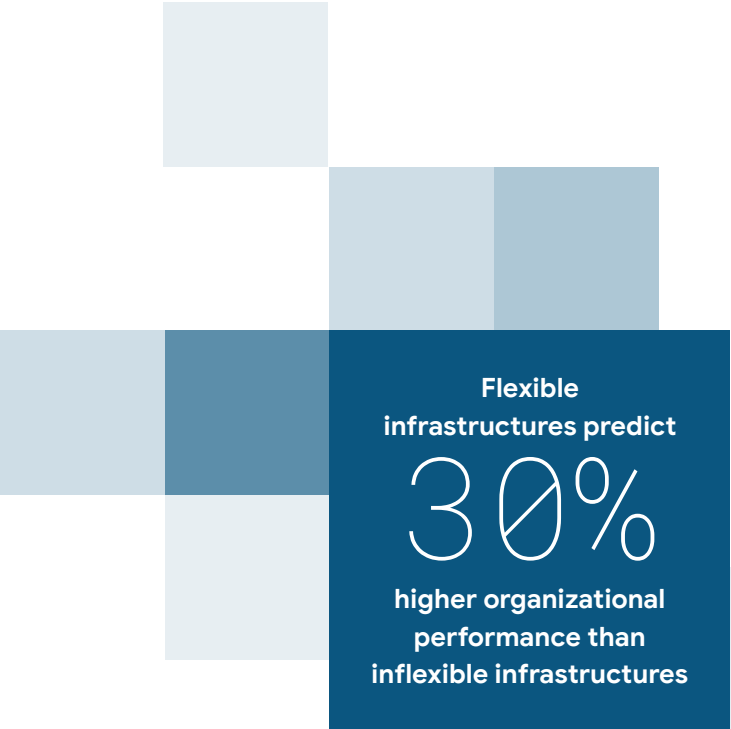
## Introduction

Throughout much of DORA's research, we've asked practitioners about their infrastructure by focusing on the essential characteristics of cloud computing— as defined by the National Institute of Standards and Technology (NIST):[1]

- On-demand self-service

- Broad network access

- Resource pooling

- Rapid elasticity

- Measured service

We have consistently seen that these five characteristics predict improved organizational performance and improved software delivery performance. This year we wanted to see if using cloud computing predicted more flexible infrastructure.

[1] NIST Special Publication 800-145: "The NIST Definition of Cloud Computing."

|  | Percentage |
| --- | --- |
| Multi-cloud | 19.6% |
| Public cloud | 51.5% |
| Hybrid cloud | 33.6% |
| On premises | 19.8% |
| Under the desk | 3.2% |
| Other | 2.5% |

\* Respondents were able to select multiple answers.

## Results

Once again, we confirmed previous findings: *how* a team uses the cloud is a stronger predictor of performance than simply *that* they use the cloud. While the use of cloud can be a powerful enabler, it doesn't automatically produce benefits. In fact, we see strong indicators that public cloud leads to *decreased* software and operational performance *unless* teams make use of flexible infrastructure. This finding further promotes the idea that simply "lifting and shifting" (the act of shifting workloads from a data center to the cloud) is not beneficial and can be detrimental.

The use of cloud computing is associated with a substantial decrease in burnout, and substantial increases in job satisfaction and productivity.

**Computing environments**
The table above shows where respondents said their primary application or service is running.

| Cloud type | Organizational performance | Team performance | Software delivery performance | Operational performance |
|---|---|---|---|---|
| Private | ⬆⬆ Very substantial increases associated with using cloud computing | ⬆⬆ Substantial increases associated with using cloud computing | ⊖ No sign of impact | ⬆⬆ Substantial increases associated with using cloud computing |
| Public | | | ⬆⬇ Substantial decreases associated with using cloud computing | ⬆⬇ Substantial decreases associated with using cloud computing |
| Hybrid | | | | ⊖ No sign of impact |
| Multi | | | ⊖ No sign of impact | ⬆⬇ Substantial decreases associated with using cloud computing |

**Simply "using cloud" provides mixed results**

As the results table shows, simply "using cloud" has either neutral or negative impacts on software delivery and operational performance. This neutral-to-negative impact is likely the result of practitioners who have taken their first step on their cloud journey, and are now faced with working in a new environment, working with new tools, and doing *some* things differently. Often, companies use cloud in the same way they did in their own data centers, only with the added complexities and cognitive burden of a new environment. Failing to adapt to this new environment doesn't improve software delivery or operational performance, but instead hurts them.

The one exception to this finding is that of operational performance within the context of private cloud.

What *does* improve software delivery and operational performance is **flexible infrastructure** which we will discuss shortly.
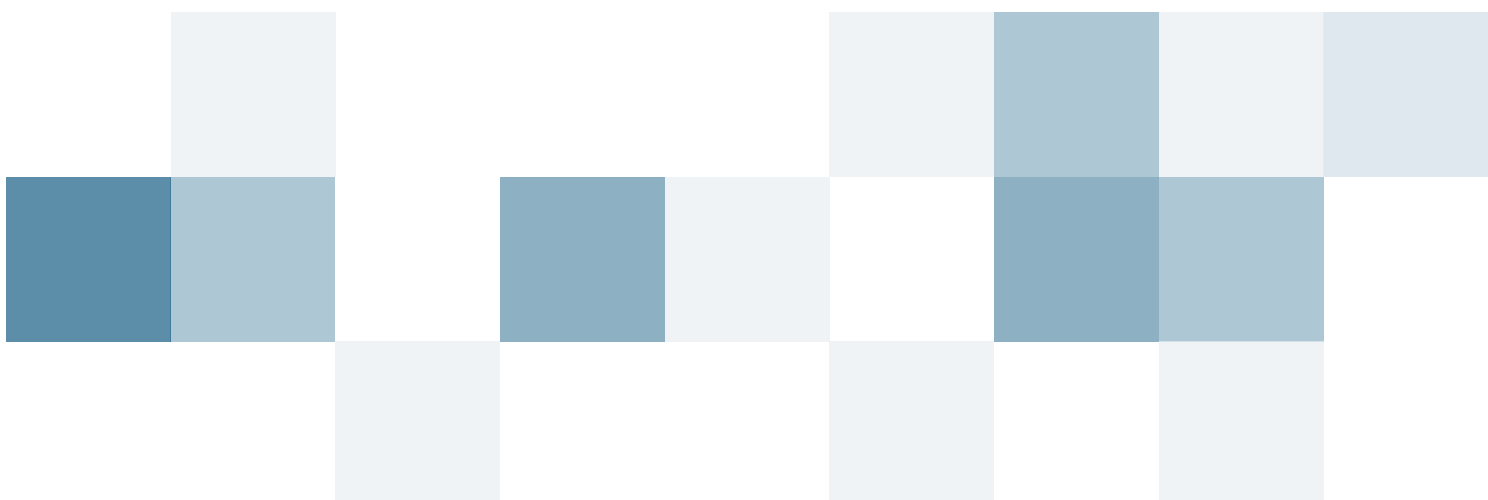
**Cloud infrastructure enables flexibility**

Using a public cloud leads to a 22% increase in
infrastructure flexibility relative to not using the cloud.
Using multiple clouds *also* led to an increase, albeit
less than a single public cloud—the obvious question
here is, why? Our data shows flexible infrastructure,
often enabled by cloud computing, is more impactful
than just using a cloud platform. For most, cloud
represents a new way of doing things and mastery
takes time. Since each cloud platform is different,
this means that as you increase cloud platforms,
you increase the cognitive burden required to
operate each platform well.

Using a public cloud
leads to an increase in
infrastructure flexibility
relative to not using
the cloud.

**Change in infrastructure flexibility relative to non-cloud users**
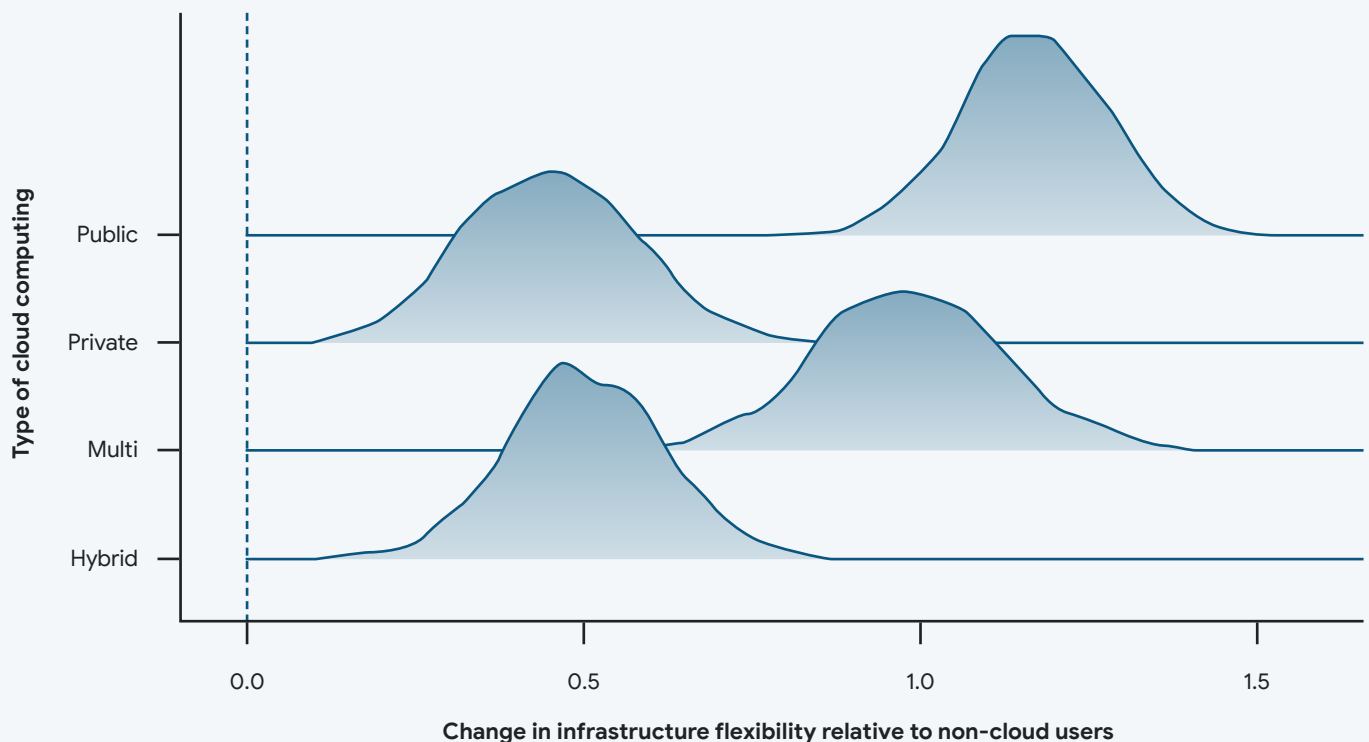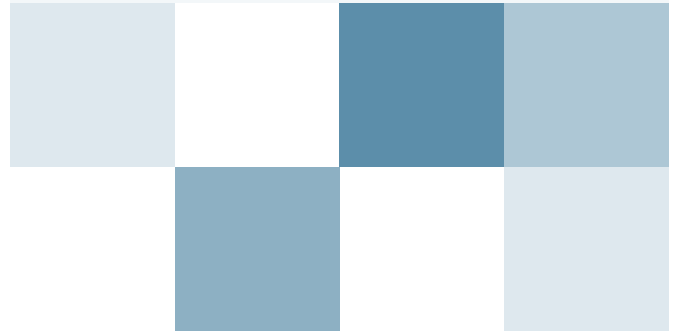
**Flexible infrastructures predict higher performance on key outcomes**

| Capability | Organizational performance | Team performance | Software delivery performance | Operational performance |
|---|---|---|---|---|
| Flexible infrastructure | ⇈ Substantial increases associated with more flexible infrastructure | ⇈ Substantial increases associated with more flexible infrastructure | ⇈ Substantial increases associated with more flexible infrastructure | ↑ Increases associated with more flexible infrastructure |

It is important to recognize that flexible infrastructure drives success in organizational performance, team performance, software delivery performance, and operational performance. Many organizations choose to lift and shift infrastructure to the cloud, and this can be a great first step, but it is just the beginning of the journey. If you do decide to lift and shift a portion of your workloads, your next step is to modernize them by refactoring to make use of flexible infrastructure.
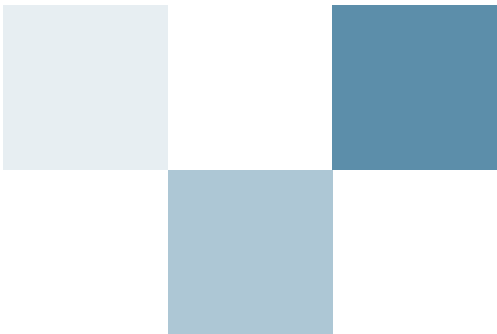
**Cloud computing has a positive impact on key outcomes through flexible infrastructure**

| Cloud type with flexible infrastructure | Organizational performance | Team performance | Software delivery performance | Operational performance |
|---|---|---|---|---|
| Private | ↑ Partially mediated by flexible infrastructure | ↑ Partially mediated by flexible infrastructure | ↑ Partially mediated by flexible infrastructure | ↑ Partially mediated by flexible infrastructure |
| Public | ⇈ Fully mediated by flexible infrastructure | ⇈ Fully mediated by flexible infrastructure | ↑ Partially mediated by flexible infrastructure | ↑ Partially mediated by flexible infrastructure |
| Hybrid | ↑ Partially mediated by flexible infrastructure | ⇈ Fully mediated by flexible infrastructure | ↑ Partially mediated by flexible infrastructure | ⇈ Fully mediated by flexible infrastructure |
| Multi | ↑ Partially mediated by flexible infrastructure | ↑ Partially mediated by flexible infrastructure | ↑ Partially mediated by flexible infrastructure | ↑ Partially mediated by flexible infrastructure |
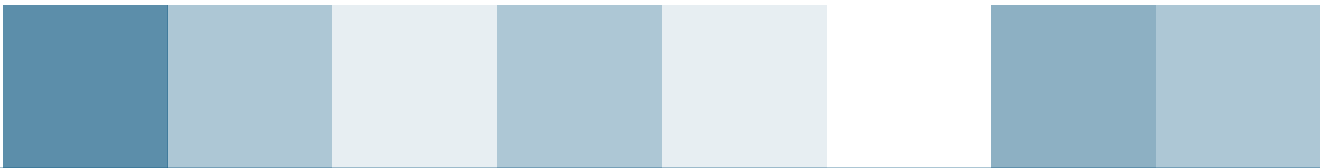
Cloud computing platforms, when used in a way that maximizes the flexible infrastructure characteristics, predict a positive impact on software delivery and operational performance. This difference in impact speaks to what most practitioners and leadership already know—simply shifting your workloads from a data center to the cloud does not bring success. The key is to take advantage of the flexible infrastructure that cloud enables.

| Infrastructure type | Outcomes |
|---|---|
| Cloud coupled with flexible infrastructure |  |
| Cloud without flexibility |  |

To maximize your potential for benefit, you must rethink how you build, test, deploy, and monitor your applications. A big part of this rethinking revolves around taking advantage of the five characteristics of cloud computing: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service.

## Cloud computing improves well-being

| Cloud type | Burnout* | | Job satisfaction | Productivity |
| --- | --- | --- | --- | --- |
| Private | ⊖ | No sign of impact | | |
| Public | ⬇⬇ | Very substantial decreases associated with using cloud computing | ⬆⬆ Substantial increases associated with using cloud computing | ⬆⬆ Substantial increases associated with using cloud computing |
| Hybrid | ⊖ | No sign of impact | | |
| Multi | ⊖ | No sign of impact | | |

*You might notice how the color scheme is flipped for burnout. This is because reducing burnout is a good thing!

The data shows that cloud computing is largely beneficial to employee well-being. We see a substantial increase in both job satisfaction and productivity, and a neutral or positive impact on burnout. Said another way, cloud doesn't have a detrimental effect on well-being even though cloud computing comes with additional cognitive burden, learning new tools, and new ways of working.

As practitioners ourselves, we hypothesize a few reasons why we are seeing this. Engineers like learning and solving problems, and enjoy working in an environment with flexible computing characteristics. Learning new technologies is not only fun, but is a great form of career development. Engineers are happier when their organization is succeeding.

Chapter 7
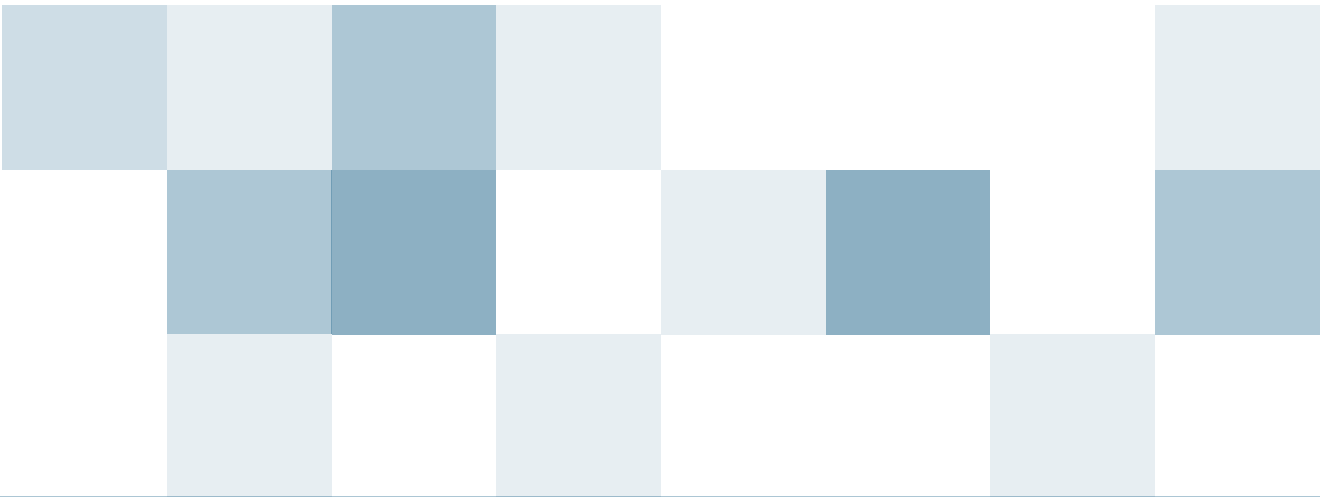
# None of this works without investing in culture

## Takeaways

Culture is a key driver of employees' well-being and organizational performance. A healthy culture can help reduce burnout, increase productivity, and increase job satisfaction. It also leads to meaningful increases in organizational performance, in software delivery and operational performance, and in team performance. A healthy organizational culture can help teams be more successful at implementing technical capabilities associated with improved outcomes.

# Introduction

Culture is hard to define. We focus on indicators that can tell us something about people's experience at work. We use Westrum's typology of organizational culture[1] because it has consistently been a strong predictor of performance. And we see this year that organizations that have a generative culture, as defined by Westrum, continue to perform well.

The following table lists aspects that we think contribute to team and organization culture.

| Aspect | Definition |
| --- | --- |
| Westrum's organizational culture | How an organization tends to respond to problems and opportunities. There are three types of culture: generative, bureaucratic, and pathological. |
| Organization stability | How stable or unstable the environment is for employees. |
| Job security | How often employees worry about their job security. |
| Flexibility | How one works, where one works, and when one works. |
| Knowledge sharing | How ideas and information spread across an organization. Team members answer questions once, and the information is available to others. People don't have to wait for answers. |
| User-centrism | A focus on the end user when developing software and a deep understanding of users' needs and goals. User signals are used to make products and services better. |
| Work distribution | Formal processes that help teams distribute burdensome tasks equitably across its members. |

The lines between cultural aspects, process capabilities, and technical capabilities are not always clear. We believe that culture emerges from practices, and practices emerge from culture. We touch on this later when discussing our findings.

[1] http://bmj.co/1BRGh5q

# What did we find and what does it mean?

**Healthy culture improves key outcomes**

Overall, a healthy culture has a positive impact on all key outcomes. We replicate previous years' findings that a generative culture drives organizational performance, software delivery performance, and operational performance. It also drives this year's new performance metric: team performance.

We found that a user-centered approach to software development leads to meaningful increases in performance. This is worth highlighting. Organizations can experience a cascade of benefits when they put the user first. User feedback helps teams prioritize projects and helps them create products and services that meet user needs. This leads to a better user experience, increased user satisfaction, and increased revenue.

We also assessed the health of an organization's culture by measuring work distribution across teams. We found that equitable work distribution benefits team and organizational performance. However, we found that equitable work distribution was associated with *lower* software delivery performance. Perhaps formal processes around work distribution slow the completion of burdensome tasks that are part of the software delivery pipeline. It's also possible that formal processes impact who within the team should take on a given task.

Another seemingly incongruent finding is that organization stability shows a small but significant

**Teams with generative cultures have 30% higher organizational performance than teams without**

decrease in software delivery performance. A potential explanation is that more established (and likely larger) organizations don't feel pressure to move as fast as newer, less established (and smaller) organizations. More established organizations might already have an established product, which gives them flexibility around the speed of their software delivery.

When information flows easily, things get done. We found that higher levels of information sharing were associated with increased software delivery performance and operational performance. When information is readily accessible and when there are few knowledge silos, people can spend time on tasks that matter instead of chasing information needed to perform those tasks.

Finally, flexible work arrangements, where employees can determine when, where, and how they work, have a beneficial impact across all performance metrics. This is particularly true for software delivery performance. Even as organizations tighten their remote-work policies, allowing employees to maintain some flexibility is likely to have a benefit.

| Aspect of culture | Effect on team performance | Effect on organizational performance | Effect on software delivery performance | Effect on operational performance |
|---|---|---|---|---|
| Westrum's organizational culture | ⬆⬆ Substantial increase | ⬆⬆ Substantial increase | ⬆⬆ Substantial increase | ⬆⬆ Substantial increase |
| Organization stability | ⬆ Minor increase | ⬆⬆ Substantial increase | ⬇ Minor decrease | ⊖ No effect |
| Job security | ⬆ Minor increase | ⊖ No effect | ⬆ Minor increase | ⬆ Minor increase |
| Flexibility | ⬆ Minor increase | ⬆ Minor increase | ⬆⬆ Substantial increase | ⬆ Minor increase |
| Knowledge sharing | ⬆ Minor increase | ⬇ Minor decrease | ⬆⬆ Substantial increase | ⬆⬆ Substantial increase |
| User-centrism | ⬆⬆ Substantial increase | ⬆⬆ Substantial increase | ⬆ Minor increase | ⬆⬆ Substantial increase |
| Work distribution | ⬆⬆ Substantial increase | ⬆⬆ Substantial increase | ⬇⬇ Substantial decrease | ⊖ No effect |

**Healthy culture improves technical capabilities**

Our findings suggest that good culture helps improve the implementation of technical capabilities. We believe the relationship between culture and technical capabilities is reciprocal: culture emerges from practices, and practices emerge from culture.

Culture is broad and hard to define, while technical capabilities are usually scoped and well-defined. This has implications for how individuals within an organization can help drive change.

For example, leaders can create incentive structures that promote a generative culture. Both leaders and individual contributors can emphasize a user-centered approach to software development. Individual contributors can help drive the implementation of technical capabilities that improve performance— trunk-based development, continuous integration, reliability practices, and loosely coupled architecture. Implementing these technical capabilities is not easy, and successfully doing so requires people to work together, to have an open mind, and to lean on and learn from each other. These are all components of a healthy culture. These teams can become examples for others within the organization, who might feel more empowered to drive change using whatever levers they have within their grasp.

Long-lasting and meaningful changes to an organization's culture come about through concurrent top-down and bottom-up efforts to enact change.

| Aspect of culture | Effect on trunk-based development | Effect on reliability practices | Effect on continuous integration | Effect on continuous delivery | Effect on loosely coupled architecture |
|---|---|---|---|---|---|
| Westrum's organizational culture | Substantial increase | Substantial increase | Substantial increase | Substantial increase | Substantial increase |
| Organization stability | Minor increase | Substantial increase | No effect | No effect | No effect |
| Job security | Minor decrease | Minor decrease | No effect | No effect | No effect |
| Flexibility | No effect | Minor decrease | Substantial increase | Minor increase | Substantial increase |
| Knowledge sharing | No effect | No effect | No effect | Minor increase | Minor increase |
| User-centrism | Substantial increase | Substantial increase | Substantial increase | Substantial increase | Substantial increase |
| Work distribution | Substantial increase | Substantial increase | Substantial increase | Substantial increase | Substantial increase |

**Healthy culture improves employee well-being**

A healthy culture leads to high levels of employee well-being by reducing burnout, increasing job satisfaction, and increasing productivity. Employee well-being is not a nice-to-have: it's foundational to an organization's overall health and success.

What happens when organizations *don't* invest in a better culture? The likelihood of burnout increases, and job satisfaction decreases. Employees become cynical and their productivity declines. Their physical and psychological health are also negatively impacted.[2, 3] Burnout is persistent; it's not something people get over after taking some time off. Burnout also increases turnover—employees leave to look for healthier work environments.[4] Therefore, alleviating burnout requires organizational changes that address its causes.

| Aspect of culture | Effect on burnout* | Effect on job satisfaction | Effect on productivity |
|---|---|---|---|
| Westrum's organizational culture | Substantial decrease | Substantial increase | Substantial increase |
| Organization stability | Substantial decrease | Substantial increase | Minor increase |
| Job security | Substantial decrease | Minor increase | Minor increase |
| Flexibility | Minor decrease | Minor increase | Minor increase |
| Knowledge sharing | Substantial decrease | Minor increase | Minor increase |
| User-centrism | Minor decrease | Substantial increase | Substantial increase |
| Work distribution | No effect | Minor increase | Minor increase |

*You might notice how the color scheme is flipped for burnout. This is because reducing burnout is a good thing!

[2] Adam Bayes, Gabriela Tavella & Gordon Parker (2021) "The biology of burnout: Causes and consequences," The World Journal of Biological Psychiatry, 22:9, 686–698. DOI: 10.1080/15622975.2021.1907713. https://doi.org/10.1080/15622975.2021.1907713

[3] Maslach C, Leiter MP. "Understanding the burnout experience: recent research and its implications for psychiatry." World Psychiatry, June 2016, 15(2), 103–11. DOI: 10.1002/wps.20311. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4911781/.1 PMID: 27265691; PMCID: PMC4911781.

[4] L.A. Kelly, et al. "Impact of nurse burnout on organizational and position turnover." Nursing Outlook, January 2021, 96–102, January–February 2021, 96–102. DOI: doi.org/10.1016/j.outlook.2020.06.008

Chapter 8

# How, when, and why who you are matters

## Takeaways

Who you are matters: we found that certain groups of respondents have different outcomes than other respondents, such as more burnout or less productivity. We have also identified specific practices you can implement to mitigate some of these negative outcomes.

## Introduction

A general phenomenon pervaded 2022's analysis: the way that work is set up might be conducive to the well-being of some, but not all.

In 2022, we found that people who identified as being underrepresented reported higher levels of burnout.[1] In this chapter, we'll see that this finding replicates, and start to address why underrepresented groups are more likely to experience burnout and what factors can help prevent this.

Further, the instability that has gripped many industries has led to questions around new hires. Organizations are concerned that it takes new employees a long time to become productive. They're looking for ways to help new employees get up to speed more quickly. We will dig into this here, too.

## What did we find and what does it mean?

**Some people are more burnt out than others**

Last year, we found that respondents who identified as women or self-described their gender, and respondents who identified as being underrepresented in any way, reported being more burnt out than respondents who identified as men and did not identify as underrepresented. These findings are consistent with a body of prior research that suggests people who are underrepresented experience a greater degree of burnout [2] and work-related stress [3] than their represented peers.

For these reasons, we were interested in examining whether disparities in burnout would be found in our data again this year, and they were. Respondents who identified as women or self-described their gender reported experiencing 6% higher levels of burnout than respondents who identified as men. Respondents who identified as being underrepresented in any way reported 24% higher levels of burnout than respondents who did not identify as being underrepresented.

[1] 2022 Accelerate State of DevOps Report. https://dora.dev/research/2022/dora-report/

[2] Sigalit Ronen and Ayala Malach Pines, "Gender Differences in Engineers' Burnout," Equal Opportunities International, November 7, 2008, https://www.emerald.com/insight/content/doi/10.1108/02610150810916749/full/html

[3] Dalessandro C;Lovell A;Tanner OC;, "Race, Marginalization, and Perceptions of Stress Among Workers Worldwide Post-2020." Sociological Inquiry, August 3, 2023, https://onlinelibrary.wiley.com/doi/pdf/10.1111/soin.12505

**Some types of work predict more burnout**

We found that aspects of the workplace that might seem neutral or beneficial, like quality documentation or a stable team, don't reduce burnout for all individuals (see Chapter 4 - Documentation is foundational). We wonder if this might be because of tasks that benefit the organization but contribute to burnout for some individuals.[4]

To understand the respondents' experience of burnout, we asked about the work that they do. We measured this in two ways, looking at:

- Specific tasks, like coding, meetings, or supporting teammates.

- Characteristics of the work, like unplanned work, its visibility, or amount of toil.

Characteristics of the work are important, because the same task might be experienced differently by different people or at different times. For example, some code reviews might be unplanned toil, and other reviews might be highly visible within your team, showcasing leadership and technical expertise.[5]

Respondents who identify as underrepresented reported doing 24% more repetitive work (toil) than those who do not identify as underrepresented. Respondents who identified as women or self-described their gender reported doing 40% more repetitive work than respondents who identified as men. These two groups of respondents also report doing more unplanned work, and work that is neither as visible to their peers nor aligned directly with their professional skill set. These findings partially explain the burnout reported by these groups.

## Non-promotable tasks

In their book *The No Club*, Babcock et al. describe a specific type of work: "A non-promotable task matters to your organization but will not help you advance your career."[6]

Here the word *advance* is used broadly, for example, increased compensation or improved marketability for other jobs.

Evidence shows that women do more of this type of work. Babcock et al. describe the reasons for this unequal distribution of work. For example, women are more likely to be asked to do this type of work, and more likely to say yes when asked as there's a social cost to saying no.

They also describe the consequences of the unequal distribution of these tasks, for example, some women:

- See negative impacts on their careers or earnings.

- Take on more hours to have an adequate volume of work that is relevant to their career.

---

[4] Linda Babcock, Brenda Peyser, Lise Vesterlund, and Laurie Weingart. *The No Club* (New York: Simon and Schuster, 2022), 17.

[5] Murphy-Hill, E. et al. "Systemic Gender Inequities in Who Reviews Code," Computer Supported Cooperative Work (2023) (to appear), https://research.google/pubs/pub52204

[6] Babcock et al., *The No Club*, 17.

**Formal processes of work distribution reduce burnout for some respondents**

We asked respondents if they have formal processes to distribute work evenly. We call this **work distribution**, and we expected to see this mitigate the burnout experienced by some respondents.

We found that work distribution did reduce burnout for respondents who identified as men and for respondents who identified as women or self-described their gender. With a high level of work distribution, the difference in burnout across genders disappeared.

We were surprised to find that work distribution had no impact on the level of burnout experienced by respondents who identify as underrepresented. This finding raises more questions: Do formal processes to distribute work evenly still result in unequal work distribution? Does "equal work" take into account the characteristics of tasks, like interruptions or visibility? And how do we mitigate other factors contributing to burnout, apart from work tasks, that might be more significant for this group?

# Fostering a culture of belonging

**Author: Dr. Jeffrey Winer,**

**Attending Psychologist, Boston Children's Hospital Assistant Professor, Harvard Medical School**

**A key finding and some context**

A key finding of this report is that individuals who define themselves as underrepresented experience much higher burnout than their colleagues. The report explored some possible reasons for this. In this section, we want to connect these findings to broader research on belongingness and associated organizational practice strategies.

Identifying as underrepresented in a group demonstrates a vulnerability to "belonging uncertainty,"[7] a well-established psychological phenomena  (see "Understanding and Overcoming Belonging Uncertainty" by Dr. Geoffrey Cohen).[8] This uncertainty (for example, "Do I belong here", "Can people like me be successful here?") is either reinforced or redefined through people's continued experiences and interpretations of those experiences. These well-established processes related to belonging uncertainty may help contextualize the finding from this report that individuals who identify as underrepresented report higher levels of burnout.

**What can organizations do?**

It is important to remember that diversity, inclusion, equity, and belonging mean different things and to achieve them, they require different, interconnected, and sustained strategies. Achieving belongingness requires true and sustained commitments.

If individuals struggle in an organization, the first question shouldn't be: "What is wrong with this individual?" The first questions should be: "Why would it make sense for someone to feel this way and what structural elements of our organization facilitate this feeling (for example, what elements keep this feeling in place or make it worse)?"

When problems are identified, changes should be at the organizational level while also providing support at the individual level—a "both and" approach. Supporting individuals to impact the systems governing an organization will allow changes to become built into the system and outlast the individual actors. Taking this systems and sustainability mindset will allow changes to be built into the institution so they outlast individual actors. This generative quality is what can allow organizations to strive towards belongingness. *Strive* is key here. Belongingness is built through sustained experience and action; it is never done, and that is why it is so fundamental to workplace health and productivity.

A number of tools exist to support organizations in this work. For example, the 2023 Surgeon General's report on the topic of loneliness identifies that social connection and belongingness are key antidotes to loneliness and burnout.[9]

---

[7] Walton GM;Cohen GL;, "A Brief Social-Belonging Intervention Improves Academic and Health Outcomes of Minority Students," Science (New York, N.Y.), accessed September 20, 2023, https://pubmed.ncbi.nlm.nih.gov/21415354/

[8] https://behavioralscientist.org/understanding-and-overcoming-belonging-uncertainty/

[9] Office of the Assistant Secretary for Health (OASH), "New Surgeon General Advisory Raises Alarm about the Devastating Impact of the Epidemic of Loneliness and Isolation in the United States," HHS.gov, May 3, 2023, https://www.hhs.gov/about/news/2023/05/03/new-surgeon-general-advisory-raises-alarm-about-devastating-impact-epidemic-loneliness-isolation-united-states.html

**New hires do struggle with productivity**

New hires (<1 year of experience on team) score 8% lower on productivity than experienced teammates (>1 year experience). Maybe this is to be expected. Starting on a new team is challenging and even if you are experienced in the role, the amount of team-specific knowledge required to get off the ground can be daunting. Further, being on a team is more than just a matter of skills and knowledge. Anecdotally, there is also a social component that is essential for productivity. Things like belonging, feeling like a contributing member, and psychological safety take time to develop.

**Is there anything that might help new hires ramp up?**

We hypothesized that organizations could help new hires in three ways:

- Providing high-quality documentation.

- Incorporating artificial intelligence into workflows, which has been shown in other research to be more helpful for inexperienced workers than experienced workers.

- Working together in person, which some have suggested could be particularly beneficial in the onboarding phase.

The results this year suggest that high-quality documentation leads to substantial impacts in productivity (see Chapter 4 - Documentation is foundational) and AI has minor benefits on an individual's productivity (see Chapter 3 - Technical capabilities predict performance). We have no reason to expect these effects wouldn't extend to new hires. When we look at the data, this is what we see: these practices do help new hires, but these practices do not help new hires more or less than everyone else. In other words, new hires don't get any special benefits from these practices.

If you're looking to help new hires and everyone else, high quality documentation is a great place to start given the substantiality and clarity of its effect on productivity. It's worth noting that new hires on teams with well-written documentation (1 standard deviation above average) are 130% as productive as new hires on teams with poorly written documentation (1 standard deviation below average).

We'll keep the discussion on return-to-office brief to avoid adding fuel to the fire. We discuss the importance of flexibility in Chapter 7 - None of this works without investing in culture. Further, our data is not experimental, and although we try to control for factors that could bias our results, the benefits of one's work arrangement are a complex and sociologically rich issue, so it is difficult to draw sharp conclusions (worth keeping in mind when reading research or thought pieces on the subject). What is clear in our data is that flexibility has a positive impact on productivity. What it isn't clear is whether where you work does.

The same story is true for new hires. We didn't see evidence of working together in person having a particular benefit to new hires. If you're trying to optimize for productivity, giving new hires flexibility in terms of how, where, and when they work, seems to be a surer bet than forcing them to be in the office. Of course, organizations are not—and probably ought not to be—optimizing solely for productivity. We also think of productivity in terms of work that leads to value, not mere output (not lines of code), and work that doesn't cause burnout or toil.

## Afterword

# Final thoughts

Thank you for participating in this year's research and reading this report. We are always looking for better ways to explore the connections between how teams work and the outcomes they are able to achieve.

The most important takeaway from our years-long research program is that teams who adopt a mindset and practice of continuous improvement are able to achieve the best outcomes.

The capabilities we've explored can be used as dials that drive outcomes. Some of those dials are within reach of individuals, while others are only accessible through coordinated effort across your entire organization. Identify which dials need adjusting for your organization, and then make investments in those adjustments.

Improvement work is never done but can create long-term success for individuals, teams, and organizations. Leaders and practitioners share responsibility for driving this improvement work.

### How will you put the research into practice?

Explore these findings in the context of your organization, teams, and services you are providing to your customers.

Share your experiences, learn from others, and get inspiration from other travelers on the continuous improvement journey by joining the DORA community at https://dora.community.

# Acknowledgments

Every year, this report enjoys the support of a large family of passionate contributors from all over the world. All steps of its production—survey question design, localization, analysis, writing, editing, and typesetting—are touched by colleagues who helped to realize this large effort. The authors would like to thank all of these people for their input, guidance, and camaraderie.

## Contributors

**Core team**

James Brookbank
Kim Castillo
Derek DeBellis
Nathen Harvey
Michelle Irvine
Amanda Lewis
Eric Maxwell
Steve McGhee
Dave Stanke
Kevin Storer
Daniella Villalba
Brenna Washington

**Editors**

Mandy Grover
Jay Hauser
Stan McKenzie
Anna Eames Mikkawi
Mike Pope
Tabitha Smith
Olinda Turner

**Survey localization**

Daniel Amadei
Kuma Arakawa
William Bartlett
Antonio Guzmán
Shogo Hamada

Yuki Iwanari
Vincent Jobard
Gustavo Lapa
Mauricio Meléndez
Jeremie Patonnier
Miguel Reyes
Pedro Sousa
Laurent Tardif
Kimmy Wu
Vinicius Xavier
Yoshi Yamaguchi

**Advisors and experts in the field**

Jared Bhatti
Lisa Crispin

Rob Edwards
Dave Farley
Steve Fenton
Dr. Nicole Forsgren
Aaron Gillies
Denali Lumma
Emerson Murphy-Hill
Harini Sampath
Robin Savinar
Dustin Smith
Jess Tsimeris
Dr. Laurie Weingart
Betsalel (Saul) Williamson
Dr. Jeffrey Winer

## Sponsors

aiven    BROADCOM SOFTWARE    CTO.ai    Deloitte.    digital.ai

GitLab    honeycomb.io    LinearB    Octopus Deploy    Liquibase

OPSERA    Qarik    SLEUTH

# Authors

**Derek DeBellis**

Derek is a quantitative user experience researcher at Google and the lead investigator for DORA. Derek focuses on survey research, logs analysis, and figuring out ways to measure concepts that demonstrate a product or feature is delivering capital-v value to people. Derek has published on human-AI interaction, the impact of COVID-19's onset on smoking cessation, designing for NLP errors, the role of UX in privacy discussions, team culture, and AI's relationship to employee well-being and productivity. His current extracurricular research is exploring ways to simulate the propagation of beliefs and power.

**Amanda Lewis**

Amanda Lewis is the DORA.community development lead and a developer relations engineer on the DORA Advocacy team at Google Cloud. She has spent her career building connections across developers, operators, product managers, project management, and leadership. She has worked on teams that developed ecommerce platforms, content management systems, observability tools, and supported developers. These connections and conversations lead to happy customers and better outcomes for the business. She brings her experience and empathy to the work that she does helping teams understand and implement software delivery and reliability practices.

**Daniella Villalba**

Daniella Villalba is a user experience researcher at Google. She uses survey research to understand the factors that make developers happy and productive. Before Google, Daniella studied the benefits of meditation training, and the psycho-social factors that affect the experiences of college students. She received her PhD in Experimental Psychology from Florida International University.

**Dave Farley**

Dave Farley is the managing director and founder of Continuous Delivery Ltd., author of *Modern Software Engineering,* and co-author of the best-selling *Continuous Delivery* book. He is one of the authors of the *Reactive Manifesto* and a winner of the Duke Award for the open source LMAX Disruptor project. Dave is a pioneer of continuous delivery, a thought leader, and an expert practitioner in CD, DevOps, test-driven development (TDD), and software design. He has a long track record in creating high-performance teams, shaping organizations for success, and creating outstanding software. Dave is committed to sharing his experience and techniques with software developers around the world, helping them to improve the design, quality, and reliability of their software. He shares his expertise through his consultancy,[1] YouTube channel,[2] and training courses.

**Eric Maxwell**

Eric Maxwell leads Google's DevOps transformation practice, where he advises the world's best companies on how to improve by delivering value faster. Eric spent the first half of his career as an engineer in the trenches, automating all the things and building empathy for other practitioners. Eric co-created Google's Cloud Application Modernization Program (CAMP), and is a member of the DORA team. Before Google, Eric spent time whipping up awesome with other punny folks at Chef Software.

**James Brookbank**

James Brookbank is a cloud solutions architect at Google. Solutions architects help Google Cloud customers by solving complex technical problems and providing expert architectural guidance. Before joining Google, James worked at a number of large enterprises with a focus on IT infrastructure and financial services.

**Dr. Jeffrey Winer**

Jeffrey P. Winer, PhD is an attending psychologist, behavioral health systems consultant, and psychosocial treatment developer within the Boston Children's Hospital Trauma and Community Resilience Center (TCRC) and an assistant professor at Harvard Medical School. With his colleagues at the TCRC, his work is primarily focused on building, testing, disseminating, and implementing culturally-responsive & trauma-informed psychosocial interventions for youth and families of refugee and immigrant backgrounds. He is co-author of the book, Mental Health Practice with Immigrant and Refugee Youth: A Socioecological Framework.[3] He has consulted with programs across the United States and Canada. Psychosocial prevention and intervention tools he has helped develop or adapt are currently used around the world. For more info, see http://www.drjeffwiner.com.

[1] https://continuous-delivery.co.uk/engineering-for-software

[2] https://www.youtube.com/@ContinuousDelivery

[3] https://www.apa.org/pubs/books/4317536

### Kevin Storer

Kevin M. Storer is a user experience researcher at Google, where he leads research directed at understanding how software development teams interact with, and through, DevOps tools. Prior to joining Google, Kevin earned his Ph.D. in Informatics from the University of California, Irvine. Across both public and private sectors, Kevin has authored high-impact publications on the topics of human-centered programming, developer experience, information behavior, accessibility, and ubiquitous computing.

### Kim Castillo

Kim Castillo is a user experience program manager at Google. Kim leads the cross-functional effort behind DORA, from overseeing its research operations to the publication of this report. Kim also works on UX research for Duet AI in Google Cloud. Prior to Google, Kim enjoyed a career in software delivery, working in technical program management and agile coaching. Kim's roots are in psycho-social research focusing on topics of extrajudicial killings, urban poor development, and community trauma and resilience in her home country, Philippines. DORA combines Kim's multiple passions in software delivery practices, psychological safety, and teal organizations.

### Michelle Irvine

Michelle Irvine is a technical writer at Google, and she leads research on the impact and production of technical documentation. Before Google, she worked in educational publishing and as a technical writer for physics simulation software. Michelle has a BSc in Physics, as well as an MA in Rhetoric and Communication Design from the University of Waterloo.

### Nathen Harvey

Nathen Harvey leads the DORA Advocacy team as a developer relations engineering manager at Google Cloud. Nathen has had the privilege of working with some of the best teams and open source communities, helping them apply the principles and practices of DevOps and SRE. He has been a co-author of the *Accelerate State of DevOps* report for the past three years. Nathen also co-edited and contributed to *97 Things Every Cloud Engineer Should Know*.
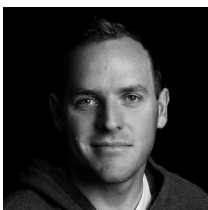
### Steve McGhee

Steve McGhee is a reliability advocate, helping teams understand how best to build and operate world-class, reliable services. Before that, he spent 10+ years as a site reliability engineer at Google, learning how to scale global systems in Search, YouTube, Android, and Google Cloud. He managed multiple engineering teams in California, Japan, and the UK. Steve also spent some time with a California-based enterprise to help them transition onto the cloud.

# Methodology

This chapter outlines how this report goes from a set of initial ideas to the report before you (with slightly more complete ideas). We hope it answers many of your questions about how this report is generated, and that it gives you a blueprint to help you embark on your own research.

## Step 1. Generate a set of outcomes we think are important to high-performing, technology-driven organizations

This is critical. Our program is based on helping guide people to the ends they care about. If we don't know where people or organizations or teams want to go, we are off to a bad start. How do we figure this out? We use a mix of qualitative research (that is, asking people what they, their teams, and their organizations want to achieve), surveys, interacting with the broader community, and many workshops. We consistently come up with outcomes like the following:

- **Organizational performance**. The organization should produce not only revenue, but value for customers, as well as for the extended community.

- **Team performance**. The ability for an application or service team to create value, innovate, and collaborate.

- **Employee well-being**. The strategies an organization or team adopts should benefit the employees—reduce burnout, foster a satisfying job experience, and increase people's ability to produce valuable output (that is, productivity).

We also hear people talk about goals like these:

- **Software delivery performance**. Teams can deploy software rapidly and successfully.

- **Operational performance**. The software that's shipped provides a reliable experience for the user.

# Step 2. Hypothesize about how, when, and why these outcomes are achieved

With the outcomes in mind from step 1, we need to hypothesize about pathways that might get teams there. This involves looking for factors that seem to have a reliable impact on an outcome. We want to say something like "Holding everything equal, x has an effect on y." This information can help practitioners make data-informed decisions about what type of changes to try.

Not only do we want to understand which pathways have an impact; we want to explore under what *conditions* these pathways might have more or less of an impact. This amounts to asking "when" and "for whom." Documentation quality, for example, has been proven to drastically reduce burnout—on average. But when we look at respondents who report as underrepresented, the opposite is true: documentation quality *increases* burnout. Understanding the conditions for these effects is vital because it is rare for any team or anyone to be average.

Further, we hypothesize about *mechanisms* that explain *why* or *how*. This year we hypothesized that people who identify as underrepresented experienced more burnout, based on results we saw last year and based on extensive literature on the topic. The first question people ask is "Why is this happening?" To try to answer this question, we hypothesized about potential mechanisms to test. For example, people who identify as underrepresented might experience more burnout *because* they take on (or are assigned) more toilsome work.

These hypotheses are then mapped out so we can construct our survey and run our analyses. Here is an example of a hypothetical model for the documentation chapter, complete with effects, conditionality, and mechanism:

**Well-being**

- Job satisfaction
- Burnout
- Productivity

**Key outcomes**

- Team performance
- Organizational performance
- Software delivery performance
- Operational performance

**Gender**

**Attenuates**

**Underrepresented**

**Attenuates**

**Knowledge sharing**

**Amplifies**

**Technical capabilities and processes**

- Trunk-based development
- Loosely coupled architecture
- Code review speed
- Continuous intergration
- AI
- Continuous delivery

**Reverses**    **Reverses**

**Work distribution**

**Documentation quality**

+    +    +    +    +

# Step 3. Hypothesize about potential confounds

If you've ever discussed data, you've probably run into a spurious correlation. You might be familiar with a website that displays many spurious correlations,[1] such as the following example:

**Per capita consumption of mozzarella cheese correlates with civil engineering docorates awarded**
Correlation: 95.86% (r=0.958648)



Data sources: U.S Department of Agriculture and National Science Foundation

— Mozzarella cheese consumption
— Engineering doctorates
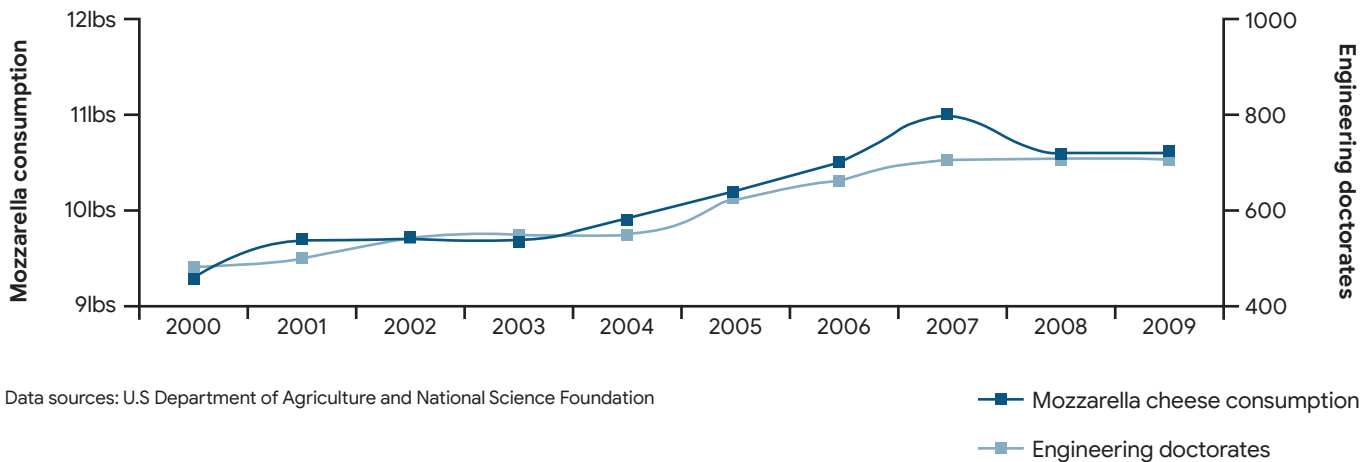
It's unlikely that there's any causal connection between engineering doctorates and mozzarella cheese consumption. Nonetheless, there is a confounding element lurking behind this relationship: time. If mozzarella cheese consumption and engineering doctorates both trend positively in the same time period they will likely have a positive correlation.

Including time in a model, or detrending the data, probably nullifies the relationship. We can draw the model like the following:



[1] https://www.tylervigen.com/spurious-correlations
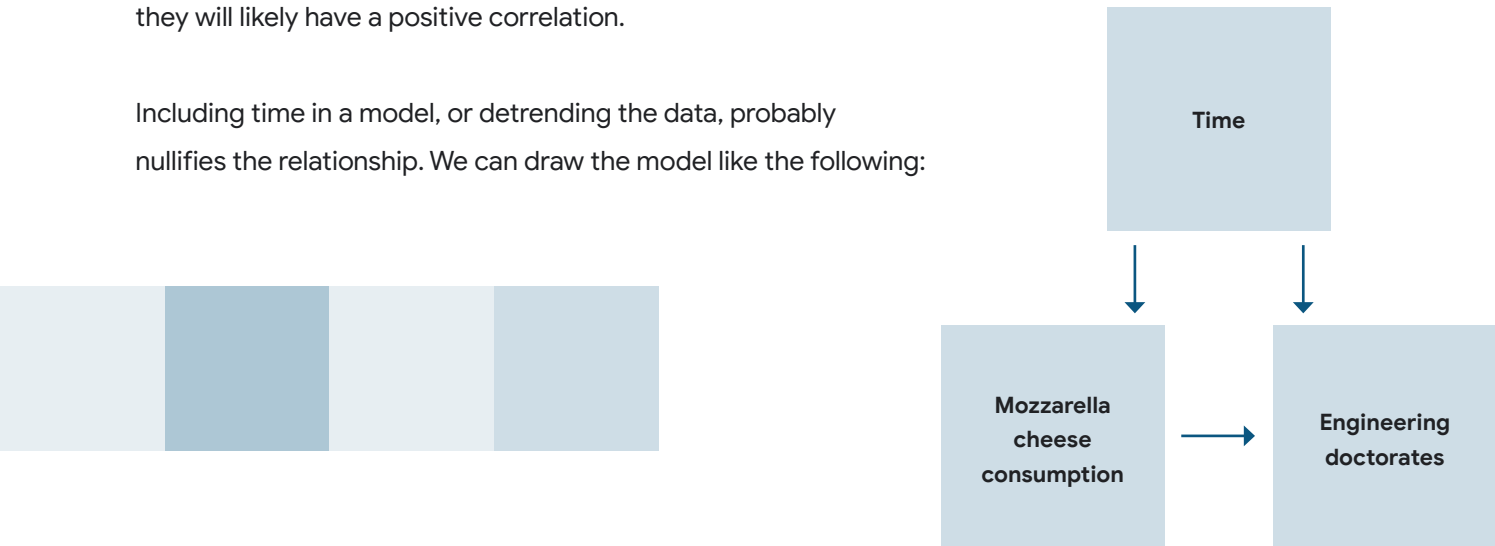
If we don't account for time (a third variable), the data might show a spurious relationship between mozzarella cheese consumption and engineering doctorates.

There are tools to help researchers with this, such as Dagitty (https://dagitty.net/dags.html). This tool lets us specify our causal model. To help us properly estimate the effect of X on Y, it tells us the implications of the model, what we need to account for, and what we ought *not* to account for. Tools like Dagitty can lead to the conclusion that correlation might not imply causation, but it most certainly implies the way someone is *thinking* about causation.

It's impossible to capture all the elements that bias researchers' estimates—think 50-hour-long surveys and omniscience. Still, we do our best to account for biasing pathways so that we can give you accurate estimates of the effects of various activities, technologies, and structures on the outcomes you care about. At the end of the day, many practitioners want to know what factors will impact these key outcomes. Models that fail to account for biases will fail to provide practitioners the guidance they need. We don't want to tell someone that mozzarella cheese increases software delivery performance—and it would be easier to make that mistake than you might imagine.

# Step 4. Develop the survey

There are three aspects to developing the survey: operationalization, experience, and localization.

## Operationalization

We want measures that adequately capture the concepts we're interested in, and that do so reliably. Translating an abstract concept into something measurable is the art of operationalization. These measures are the ingredients at the base of all the analysis. If our measures are not giving us clear signals, how can we trust the rest of the analysis? How do we measure a concept as elusive as, for example, productivity? What about burnout or operational performance?

First, we look to the literature to see if there are successful measures that already exist. If we can use previously validated measures in our survey, we gain a bridge from the survey to all the literature that has amassed around that question. Our ongoing use of Westrum's *Typology of Organizational Cultures* is an example of us reusing previously validated measures.

However, many concepts haven't previously been validated for the space we do research in. In that case, we're doing qualitative research to untangle how people understand the concept and we're looking through the more philosophical literature on the intricacies of the concept.

## Survey experience

We want the survey to be comprehensible, easy, no longer than necessary, and broadly accessible. These are difficult goals, given all the questions we want to ask, given the technical understanding required in order to answer these questions, and given the variation in nomenclature for certain practices. We do remote, unmoderated evaluations to make sure the survey is performing above certain thresholds. This requires doing multiple iterations.

## Localization

People around the world have responded to our survey every year. This year we worked to make the survey more accessible to a larger audience by localizing the survey into English, Español, Français, Português, and 日本語. This was a grassroots effort, led by some incredible members of the DORA community. Googlers all over the world contributed to this effort, as well as a partner in the field—many thanks to Zenika (https://www.zenika.com) for our French localization. We hope to expand these efforts and make the survey something that is truly cross-cultural.

# Step 5. Collect survey responses

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

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

We use the *panel approach* to supplement the organic channel. Here we try to recruit people who are traditionally underrepresented in the broader technical community, and we try to get adequate responses from certain industries and organization types. In short, this is where we get some control over our recruitment—control we don't have with the organic approach. The panel approach also allows us to simply make sure that we get enough respondents, because we never know if the organic approach is going to yield the responses necessary to do the types of analyses we do.

# Step 6. Analyze the data

There are three steps at the heart of the analysis: data cleaning, measurement validation, and model evaluation.

**Data cleaning**

The goal of data cleaning is to increase the signal-to-noise ratio. For various reasons, some responses are noise. Some response patterns can indicate that the person is distracted, speeding through the survey, or not answering in good faith. This is very rare in our data, but it happens. Responses that show signs of misrepresenting a person's actual experiences are excluded from our analysis because they are noisy. The challenge when getting rid of noise is making sure we don't get rid of the signal, especially in a biased manner or in a way that validates our hypotheses. For example, if we conclude that no one could be high on a certain value and low on another, we might exclude that respondent—causing the data to be more aligned with our beliefs, and increasing the odds that our hypotheses pan out.

## Measurement validation

At the beginning of this report, we talk about the concepts we try to measure. There are a lot of different language games we could partake in, but one view is that this measure of a concept is called a variable.[2] These variables are the ingredients of the models, which are the elements included in our research. There are two broad ways to analyze the validity of these measures: internally and externally.

To understand the internal validity of the measure, we look at what we think indicates the presence of a concept. For example, quality documentation might be indicated by people using their documentation to solve problems.

A majority of our variables consist of multiple indicators because the constructs we're interested in appear to be multifaceted. To understand the multifaceted nature of a variable, we test how well the items we use to represent that construct gel. If they gel well (that is, they share a high level of communal variance), we assume that something underlies them—such as the concept of interest.

Think of happiness, for example. Happiness is multifaceted. We expect someone to feel a certain way, act a certain way, and think a certain way when they're happy. We assume that happiness is underlying a certain pattern of feelings, thoughts, and actions. Therefore, we expect certain types of feelings, thoughts, and actions to emerge together when happiness is present. We would then ask questions about these feelings, thoughts, and actions.

We would use confirmatory factor analysis to test whether they actually do show up together.

This year we used the lavaan R package to do this analysis.[3] Lavaan returns a variety of fit statistics that help us understand whether constructs actually represent the way people answer the questions. If the indicators of a concept don't gel, the concept might need to be revised or dropped because it's clear that we haven't found a reliable way to measure the concept.

The external validity of a construct is all about looking at how the construct fits into the world. We might expect a construct to have certain relationships to other constructs. Sometimes we might expect two constructs to have a negative relationship, like happiness and sadness. If our happiness measure comes back positively correlated with sadness, we might question our measure or our theory. Similarly, we might expect two constructs to have positive relationships, but not strong ones. Productivity and job satisfaction are likely to be positively correlated, but we don't think they're identical. If the correlation gets too high, we might say it looks like we're measuring the same thing. This then means that our measures are not calibrated enough to pick up on the differences between the two concepts, or the difference we hypothesized about isn't actually there.

---

[2] Moore, Will H., and David A. Siegel. A mathematics course for political and social research. Princeton University Press, 2013.

[3] Rosseel, Y. "lavaan: An R Package for Structural Equation Modeling," Journal of Statistical Software, 48(2), 2012. 1–36. https://doi.org/10.18637/jss.v048.i02

## Model evaluation

In steps 2 and 3, we built hypothetical models. After step 6 has given us clean data, we see how well those models fit the data. This year we adopted a Bayesian approach to be able to understand how plausible various hypotheses are given the data, instead of how likely the data is given the null hypothesis (that is, no effect is present). The main tools we use in R are blavaan[4] and rstanarm.[5] We can test the probability that an effect is substantial or dramatic and not simply minor. For evaluating a model, we go for parsimony. This amounts to starting with a very simplistic model and adding complexity until the complexity is no longer justified. For example, we predict that organizational performance is the product of the interaction between software delivery performance and operational performance.

Our simplistic model doesn't include the interaction:

Organizational performance ~ Software delivery performance + Operational performance

Our second model adds the interaction:

Organizational performance ~ Software delivery performance + Operational performance + Software delivery performance x Operational performance

Based on the recommendations in "Regression and other stories"[6] and "Statistical Rethinking",[7] we use leave-one-out cross-validation (LOOCV) and Watanabe–Akaike widely applicable information criterion[8] to determine whether the additional complexity is necessary.

[4] Merkle, Edgar C., and Yves Rosseel. "blavaan: Bayesian structural equation models via parameter expansion," arXiv preprint, 2015. https://doi.org/10.48550/arXiv.1511.05604

[5] Goodrich, Ben, Jonah Gabry, Imad Ali, and Sam Brilleman. "rstanarm: Bayesian applied regression modeling via Stan." R package version 2, no. 1 (2020).

[6] Gelman, Andrew, Jennifer Hill, and Aki Vehtari. Regression and Other Stories (Cambridge University Press, 2020).

[7] McElreath, Richard. Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC, 2018.

[8] VVehtari, Aki, Andrew Gelman, and Jonah Gabry. "Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC." Statistics and Computing, 27, 2017. 1413-1432. https://doi.org/10.1007/s11222-016-9696-4
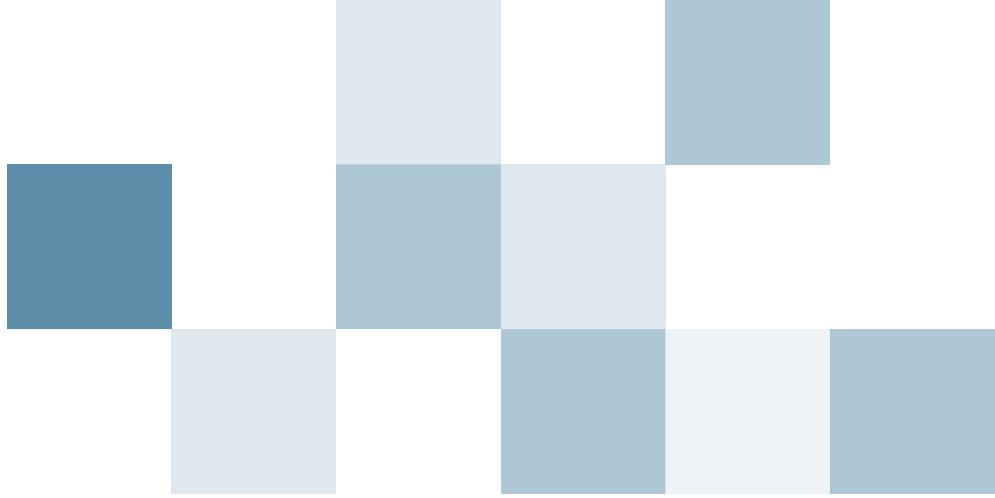
# Step 7. Report findings

We then reviewed these results as a team. This year, we spent a few days together in Boulder, Colorado, synthesizing the data with the experiences of subject-matter experts. We did this for every chapter of the report, hypothesis by hypothesis. Data interpretation always has the risks of spin, speculation, anecdotes, and leaps. These risks were mitigated by having multiple people with diverse backgrounds in a room that encouraged questioning, divergence, unique perspectives, and curiosity.[9]

With the results in hand, the report authors retreated to their respective corners of the world and wrote. Throughout the writing process, editors and subject-matter experts were consulted. Having these perspectives was vital in helping us communicate our ideas. The person responsible for analyzing this data was responsible for making sure that nothing we said deviates from what the data says.

These chapters were bundled together into a cohesive design by our talented design partners, BrightCarbon.[10]

# Step 8. Synthesize findings with the community

We count on community engagement to come up with ways both to leverage and to interpret these findings. We try to be particular in our recommendations, but in the end, there are innumerable implementations a team could try based on the results we uncover. For example, loosely coupled architecture seems to be a beneficial practice based on the outcomes we measure. But there surely isn't just a single way to establish a loosely coupled architecture. Generating and sharing approaches as a community is the only way to continually improve. Our map of the world is an interpretation and abstraction of the territory and context in which you, your team, and your organization operate.

To participate in DORA's global community of practice, visit the DORA Community site (https://dora.community).

---

[9] Stasser, G., & Titus, W. (1985). "Pooling of unshared information in group decision making: Biased information sampling during discussion." Journal of Personality and Social Psychology, 48(6), 1985. 1467–1478. https://doi.org/10.1037/0022-3514.48.6.1467

[10] https://www.brightcarbon.com/

# Demographics and firmographics

## Who took the survey

For nearly a decade, the DORA research program has been researching the capabilities, practices, and measures of high-performing, technology-driven organizations. We've heard from more than 36,000 professionals working in organizations of every size and across many different industries. Thank you for sharing your insights! This year, nearly 3,000 working professionals from a variety of industries around the world shared their experiences to help grow our understanding of the factors that drive high-performing, technology-driven organizations.

This year's demographic and firmographic questions leveraged research done by Stack Overflow. Over 70,000 respondents participated in the 2022 Stack Overflow Developer Survey.[1] That survey didn't reach every technical practitioner, for a myriad of reasons, but is about as close as you can get to a census of the developer world. With a sense of the population provided from that, we can locate response bias in our data and understand how far we might want to generalize our findings.

That data and the demographic and firmographic questions asked in the Stack Overflow Developer Survey are well-crafted and worth borrowing. Relative to the Stack Overflow Developer Survey, our sample set includes a higher proportion of women and disabled participants and participants who work in larger organizations. Our sample set is similar to Stack Overflow's in terms of race and ethnicity.

This year's survey saw a 3.6x increase in the number of organic respondents compared to 2022.

**This year's survey saw a**

# 3.6x

**increase in the number of organic respondents compared to 2022**

[1] https://survey.stackoverflow.co/2022#overview

# Demographics

## Gender

Relative to 2022, this year's sample had a smaller proportion of women respondents (12% vs. 18%).

| Gender | % of respondents |
|---|---|
| Prefer not to say | 3% |
| Or, in your own words | 2% |
| Woman | 12% |
| Man | 81% |

## Disability

We identified disability along six dimensions that follow guidance from the Washington Group Short Set.[2] This is the fifth year we have asked about disability. The percentage of people with disabilities decreased from 11% in 2022 to 6% in 2023.

| Disability | % of respondents |
|---|---|
| None of the disabilities applied | 87% |
| Yes | 6% |
| Prefer not to say / did not respond | 7% |

## Underrepresented

Identifying as a member of an underrepresented group can refer to race, gender, or another characteristic. This is the sixth year we have asked about underrepresentation. The percentage of people who identify as underrepresented has decreased slightly from 19% in 2022 to 15% in 2023.

| Underrepresented | % of respondents |
|---|---|
| No | 77% |
| Yes | 15% |
| Prefer not to respond | 7% |

[2] https://www.washingtongroup-disability.com/question-sets/wg-short-set-on-functioning-wg-ss/
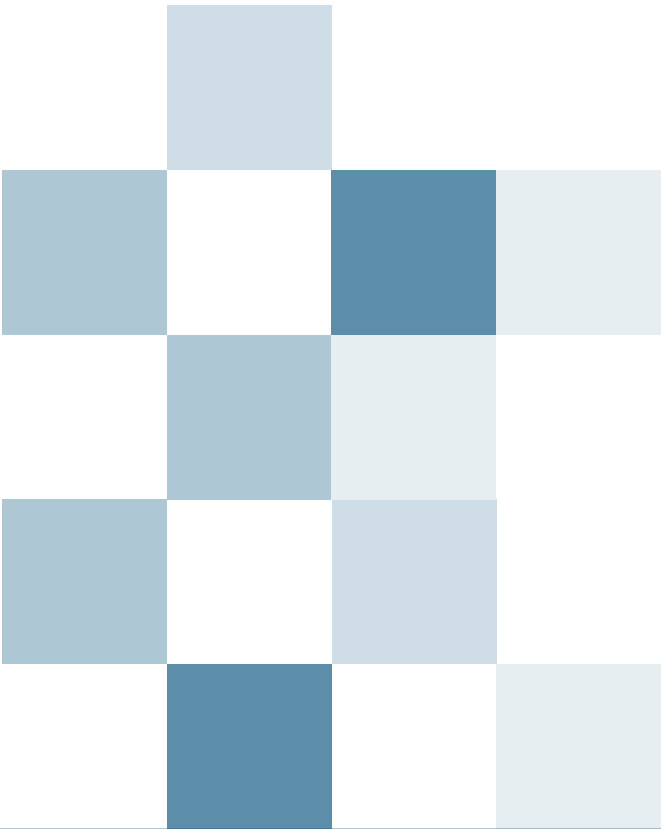
## Race and ethnicity

We adopted the question from the 2022 Stack Overflow Developer's survey.[3] As noted earlier, our sample set is similar with one notable deviation: we have a lower proportion of Europeans.



[3] https://survey.stackoverflow.co/2022#overview
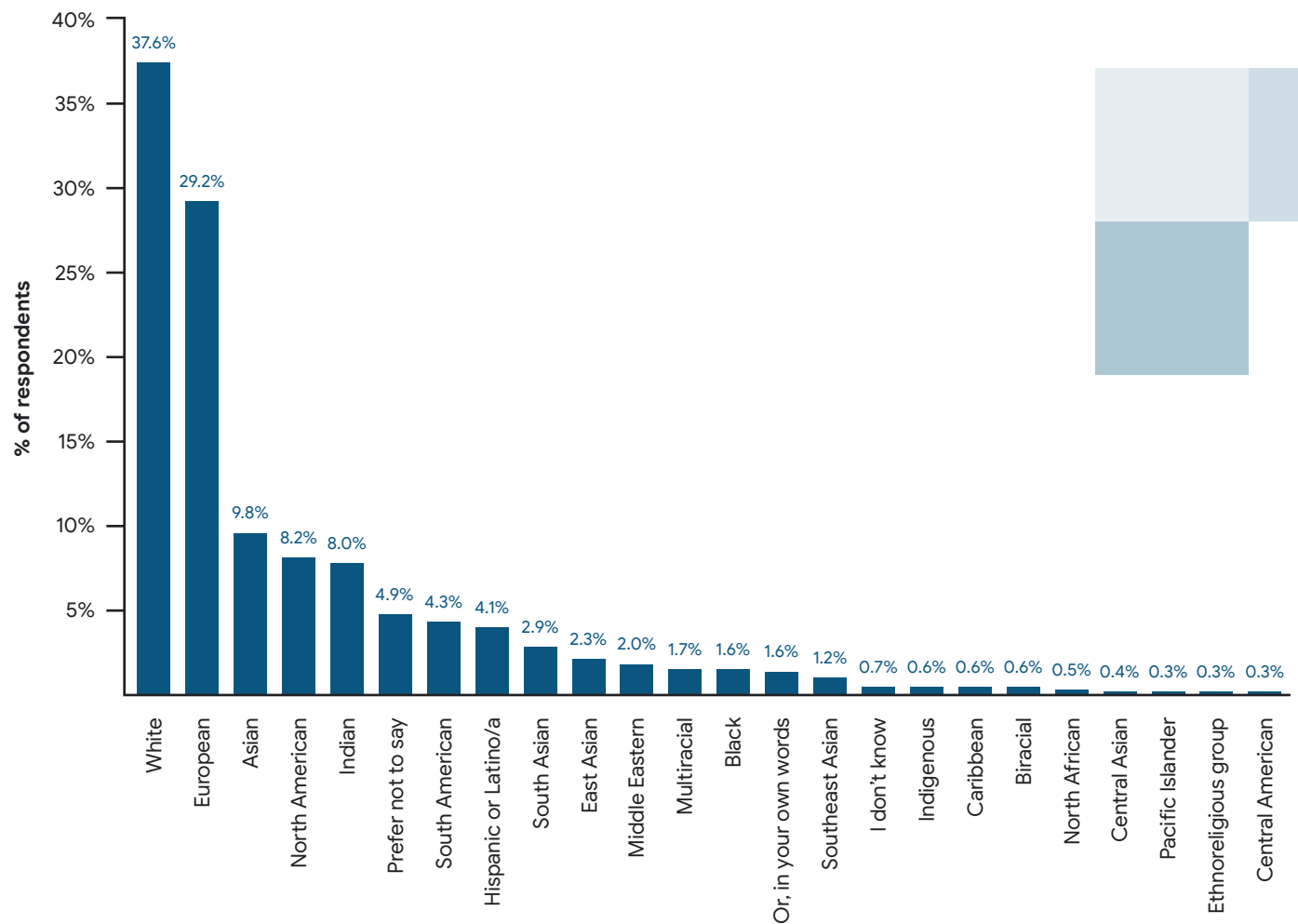
# Years of experience

**Work experience**

We wanted to understand how long someone has been working. Here we asked, "How many years of working experience do you have?" In sum, we're gathering data from a group of fairly experienced practitioners. That is, 50% of our respondents had 15 years or more of experience. 25% of respondents had more than 22 years of experience. 25% of our respondents had less than 9 years of experience. In hindsight, it isn't obvious what someone counts as "work".



**Lower 25%**
<= 9 years

**Interquartile**
(middle 50%)

**Upper 25%**
>= 22 years

% of respondents

Years of work experience

Dashed line = median
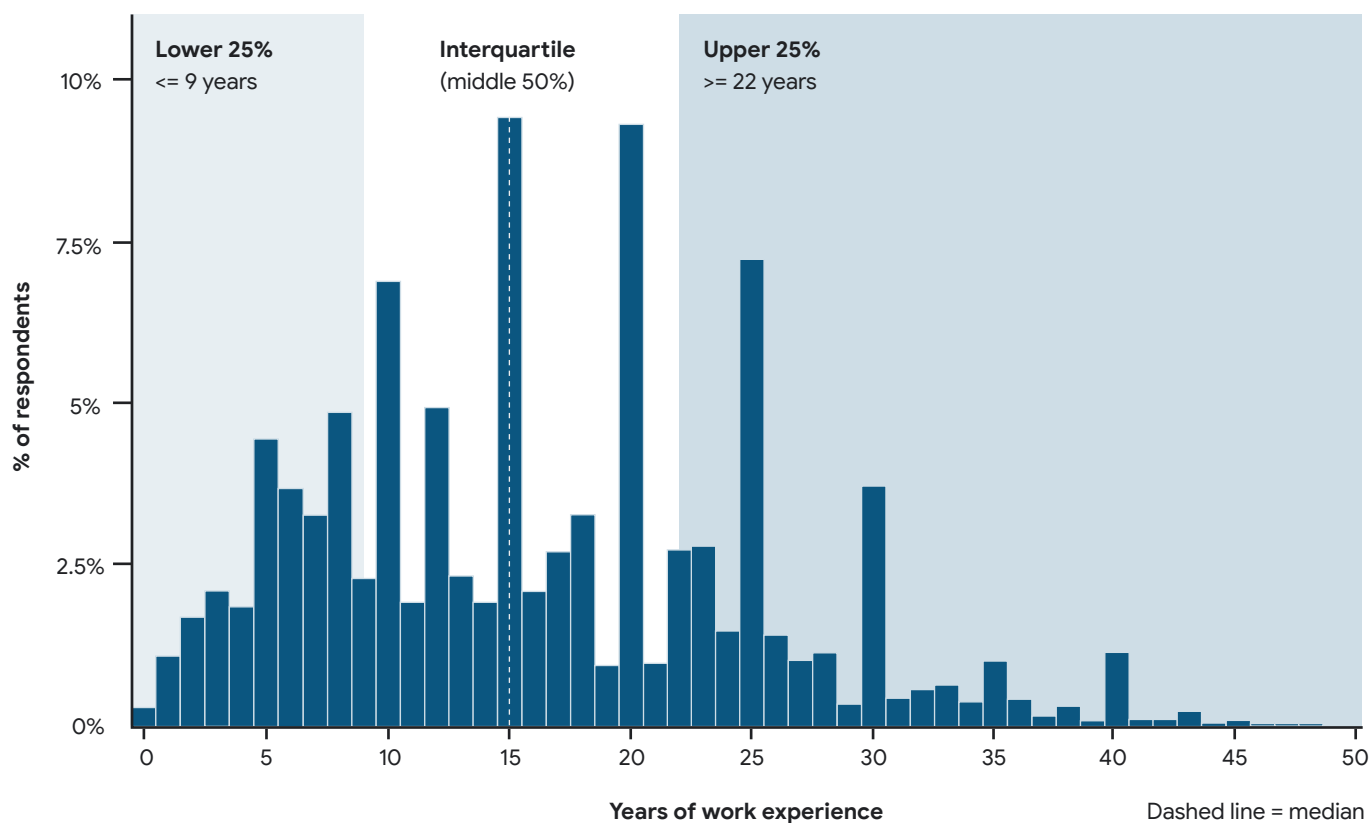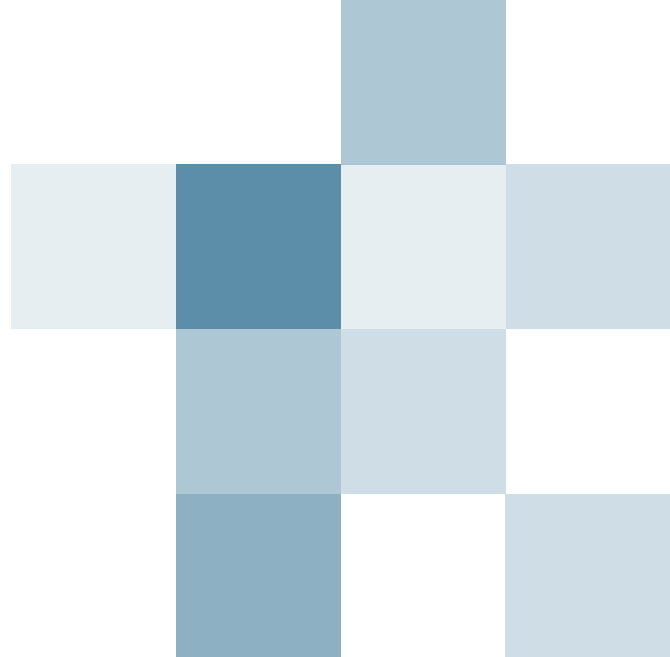
## Experience on team

Sometimes working on a new team feels like starting over, resetting a bit. We wanted to look into this so we asked, "How many years have you worked on the team you're currently on?" Despite work experience seeming high, many respondents are new to their teams. 50% of respondents have been on their new team less than 3 years. 25% have been on their team for less or equal to 1.5 years. Only 25% of our respondents have been on their team for 5 years or more. Does this reflect a mentality of continuous improvement present in our respondents? Does this reflect the flux and instability of the economy?

**Lower 25%**
<= 1.5 years

**Interquartile**
(middle 50%)

**Upper 25%**
>= 5 years

% of respondents

Years of experience on team

Dashed line = median
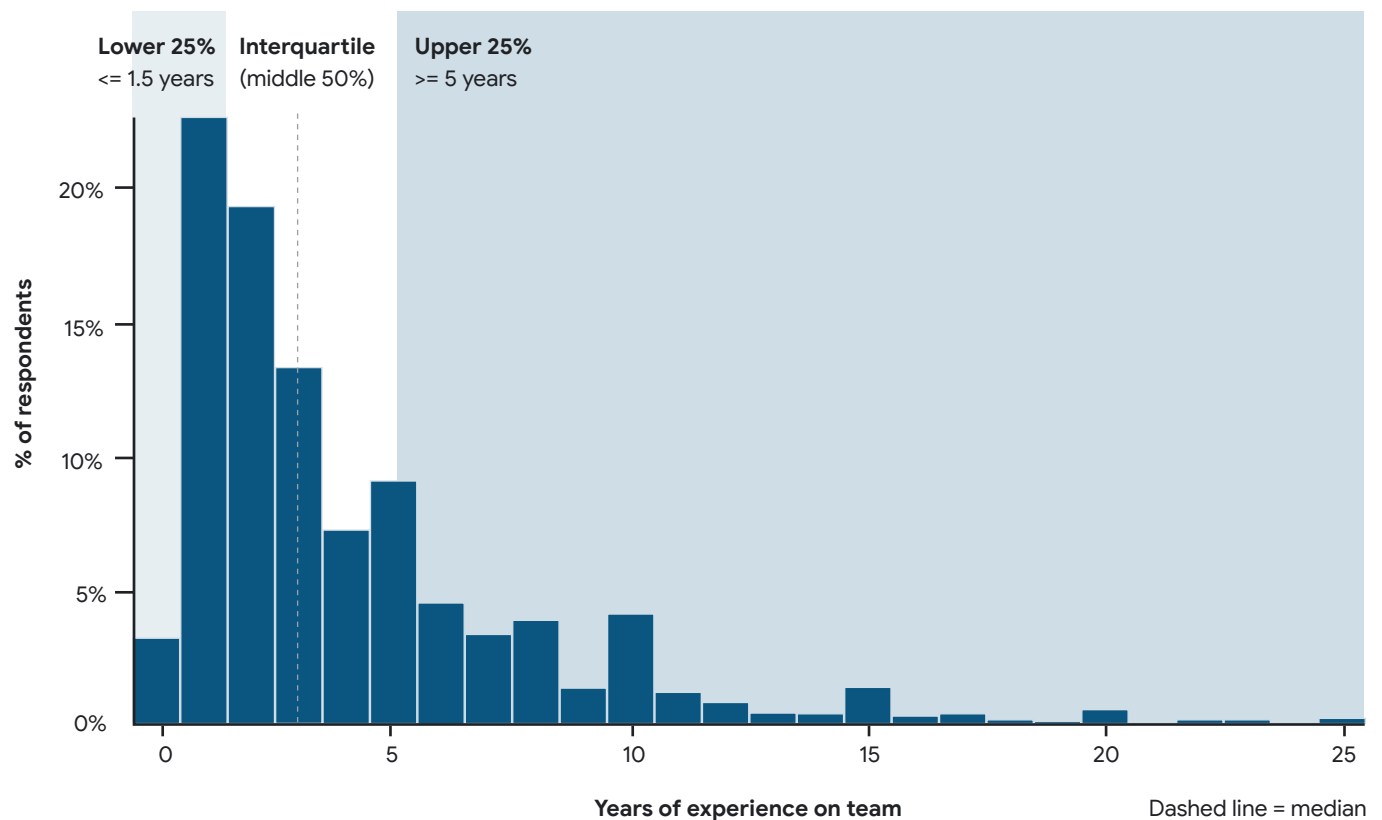
## Role

72% of respondents consist of individuals who either work on development or engineering teams (30%), work on DevOps or SRE teams (18%), work on IT ops or infrastructure teams (8%), or are managers (16%). In 2022, individuals in those roles made up 85% of respondents. The decrease in respondents from those four roles suggests that we were able to reach more individuals in different roles. The proportion of IT ops or infrastructure teams (8%) is back to 2021 levels (9%) after inflecting in 2022 (19%).



Bar chart — % of respondents by role:

- Development or engineering: 30.5%
- DevOps or SRE: 18.1%
- Manager: 16.4%
- IT operations or infrastructure: 7.6%
- C-level executive: 6.2%
- Platform engineer: 4.2%
- Consultant, coach, or trainer: 3.3%
- Other: 3.0%
- Product management: 2.5%
- No response: 2.5%
- Quality engineering or assurance: 1.1%
- Professional services: 1.0%
- Informational security: 1.0%
- Student: 0.5%
- User experience design or research: 0.4%
- Sales or marketing: 0.4%
- Sales engineering: 0.3%
- Do not belong to any department: 0.3%
- Release engineering: 0.3%
- Prefer not to answer: 0.3%
- Network operations: 0.3%

# Firmographics

**Industry**

**% of respondents**

| Industry | % |
|---|---|
| Technology | 36.6% |
| Financial services | 13.7% |
| Consumer | 8.4% |
| Other | 6.6% |
| Industries manufacturing | 5.8% |
| Healthcare pharmaceuticals | 5.7% |
| Telecommunications | 4.2% |
| Media entertainment | 4.2% |
| Government | 3.9% |
| Education | 3.3% |
| Energy | 2.3% |
| N/A | 2.3% |
| Insurance | 2.2% |
| Nonprofit | 1.0% |

**Number of employees**

How many employees work at your organization?

**% of respondents**

| Employees | % |
|---|---|
| 10,000 or more | 21.4% |
| 1,000 to 4,999 | 18.5% |
| 100 to 499 | 17.8% |
| 20 to 99 | 13.3% |
| 500 to 999 | 10.5% |
| 5,000 to 9,999 | 7.3% |
| 2 to 9 | 3.3% |
| 10 to 19 | 3.2% |
| 1 to 4 | 2.4% |
| N/A | 2.4% |

## Country

We are always thrilled to see people from all over the world participate in the survey. Thank you all!



Canada **5%**
Netherlands **3%**
**1%** Norway
**1%** Sweden
United Kingdom of Great Britain and Northern Ireland **11%**
**1%** Denmark
**1%** Poland
Spain **2%**
**5%** Germany
Portugal **1%**
**1%** Italy
**3%** France
**3%** Japan
**28%** United States of America
India **8%**
**3%** Brazil
Australia **3%**
New Zealand **1%**
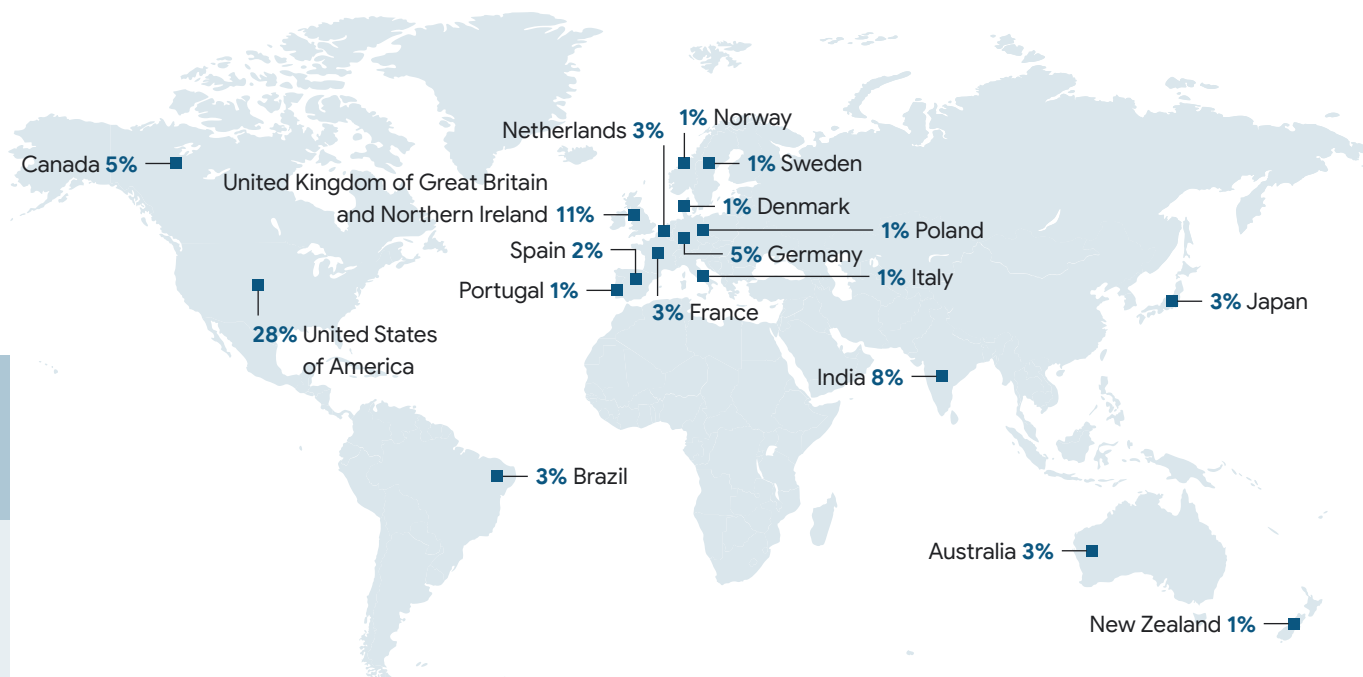
| Country | | | | | |
|---|---|---|---|---|---|
| USA | Denmark | Lithuania | Tunisia | Bangladesh | Guatemala |
| UK | Switzerland | Thailand | Uruguay | Dominican Republic | Honduras |
| India | Austria | Hungary | Afghanistan | Ghana | Latvia |
| Canada | Kenya | Israel | Algeria | Hong Kong (S.A.R.) | Lebanon |
| Germany | South Africa | Viet Nam | Egypt | Kazakhstan | Luxembourg |
| Australia | Argentina | UAE | Estonia | Myanmar | Maldives |
| Brazil | Czech Republic | Bulgaria | Iceland | Saudi Arabia | Malta |
| Not applicable | Belgium | Croatia | Iran | Somalia | Mauritius |
| Netherlands | Colombia | Ecuador | Nigeria | Sudan | Mongolia |
| Japan | Finland | Indonesia | Peru | Uganda | Morocco |
| France | Ireland | Philippines | Slovakia | Albania | Nepal |
| Spain | China | Armenia | Slovenia | Bahamas | Qatar |
| Sweden | Romania | Georgia | South Korea | Belarus | The former Yugoslav Republic of Macedonia |
| Italy | Singapore | Greece | Sri Lanka | Bolivia | |
| New Zealand | Mexico | Malaysia | Andorra | Cambodia | Trinidad and Tobago |
| Poland | Turkey | Pakistan | Angola | Costa Rica | United Republic of Tanzania |
| Norway | Ukraine | Russian Federation | Antigua and Barbuda | Djibouti | |
| Portugal | Chile | Serbia | Bahrain | El Salvador | Zimbabwe |

# Work arrangement

## Employment status

88% of the respondents are full-time employees. 10% of the respondents are contractors. Some contractors report vastly different experiences than full-time employees.

| Contract | % of respondents |
|---|---|
| Full-time employee | 88% |
| Full-time contractor | 8% |
| Part-time employee | 2% |
| Part-time contractor | 2% |

*For the primary application or service you work on, what best describes your employment status with the organization that owns the application or service?

The different experience might stem from how they fit into the team. Some contractors report being embedded in the team they work with. This means they work closely with team members every day and consider the difference between themselves and a full-time employee to be negligible. 70% of contractor employee respondents either strongly agree or agree with the statement that they are embedded on their team.

## Location

The response pattern this year indicates that, despite return-to-office pushes, working from home is still a reality for many workers. Nearly 33% of respondents work almost exclusively from home (less than 5% of time in the office). 63% of respondents work from home more than they work from the office. For the remaining respondents, hybrid work might be the most common arrangement. This is suggested by 75% of respondents spending less than 70% of their time in the office. There are not many people with a strong attachment to the office. Only 9% of respondents are in the office more than 95% of the time.
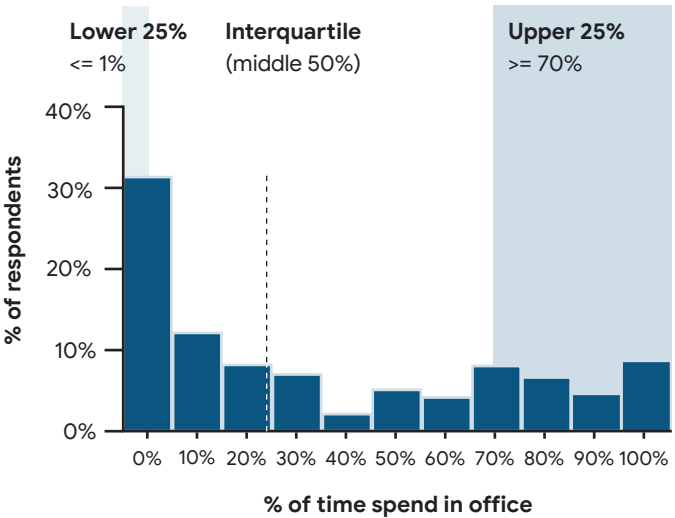
# The models

## Introduction

Traditionally, we created one giant model. This year we decided to break it down into multiple models for the following reasons:

- Huge models can become unwieldy, fast. Every added variable changes the way the model functions. This can lead to inaccurate estimates and makes it difficult to locate the reason for a change.

- We created our hypotheses section-by-section this year. Thus, it makes sense to just create a model for each section.

- It isn't obvious what the benefit of a giant model is in estimating the effect of $X$ on $Y$. To understand the impact of $X$ on $Y$, we used directed acyclic graphs to help understand what covariates we should and shouldn't include in the model.

- The number of hypotheses we addressed this year would make it very difficult for the reader to make sense of the giant model. Imagine combining all the visualizations below into one visualization.

# How do I read these diagrams?

Once you learn how to read these diagrams, you'll find them to be efficient tools for conveying a lot of information.

| | |
|---|---|
| **Variable** | Variable is a concept that we tried to measure (for example documentation quality). |
| **Variable Category**<br>• Variable<br>• Variable<br>• Variable<br>• Variable | A variable category is simply to show that we think of these as a category, but has nothing to do with the analysis. That is, we did statistically evaluate if this is a higher-order construct. |
| **+** →→→ | A positive effect, which simply means increases, not that it is good. |
| **-** →→→ | A negative effect, which simply means decreases, not that it is bad. |
| - - - → | A hypothesized effect that the data did not substantiate. |
| →→→ | Part of the mediation pathway that we explicitly analyzed. |

**Warning: the models are general summations!**

We categorize some variables together for ease of reading. This categorization strategy makes it possible for the arrow going to a variable category, from a variable category, or both, to be the general pattern of results, but it might not be true for every variable in the category. For example, knowledge sharing has a positive impact on most key outcomes. Therefore, we would draw an arrow with a plus symbol (+) on it from knowledge sharing to the key outcomes variable category. Knowledge sharing, however, doesn't have a positive impact on software delivery performance. To get into the details, please visit the relevant chapters.

**Moderation example**

Moderation is a tough concept to grasp in statistics, but in the real world, moderation amounts to saying, "it depends." Let's do a quick example to clarify the concept of moderation in the context of this report.

In season 3 of Curb Your Enthusiasm, Larry David says, "I don't like talking to people I know, but strangers I have no problem with." This is something that provides us with a quick diagram to discuss:

# "I don't like talking to people I know, but strangers I have no problem with."

Larry David
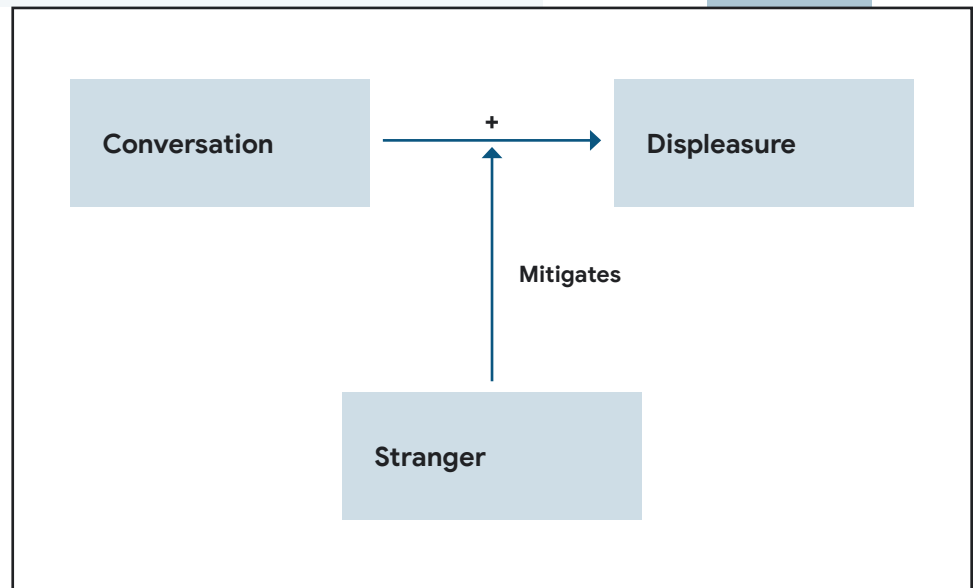


This diagram shows that, for Larry, conversation has a positive impact on displeasure. Positive here simply means increase, not that it is necessarily a good thing. This is demonstrated by the solid black line between conversation and displeasure with the arrow pointing to displeasure. This arrow suggests that we believe the causal flow is from conversation to displeasure. From what we can tell, conversations tend to cause Larry displeasure.

The second thing to note is that stranger (here to represent the boolean stranger yes / no) doesn't point to another variable. Instead, it points to an effect,

an arrow. This means we think that stranger modifies not a variable, but an effect. That is why we draw the arrow from stranger to another arrow, not to another variable. We're saying that whether or not Larry is talking to a stranger impacts the effect of conversation on displeasure. Put differently, we're saying the effect of conversation on displeasure depends on whether the person Larry is conversing with is a stranger. When the person is a stranger, the effect of conversation is something Larry "has no problem with." We might say that strangers mitigate the displeasure Larry feels while conversing.

**There are a few different ways something might moderate something else:**

- **Amplifies**—make positive effects more positive and negative effects more negative.

- **Attenuates**—weakens the effect.

- **Mitigates**—make positive effects less positive and negative effects less negative.

- **Reverses**—make positive effects negative and make negative effects positive.

- **Modifies**—sometimes the effect simply changes, but the pattern can't be summed up nicely in one word. This often happens with categorical variables as causes. For example, certain industries might behave differently under different conditions, like economic fluctuations.

### Mediation example

Like moderation, we think about mediation a lot. At the root of mediation is *why* or *how*. Why does the sun cause a plant to grow taller? Why does eating strawberries make me taste sweetness? How does pressing an accelerator pedal make my car move? How do pain relievers reduce discomfort? We can test for mediation in statistics.[1] This amounts to us being able to say, "it looks like the effect of *X* on *Y* is explained or partially explained by *M*." For example, the effect of the sun on a plant's height is explained by photosynthesis.

| Sun | → + → | Photosynthesis | → + → | Plant growth |
|-----|-------|----------------|-------|--------------|

[1] https://lavaan.ugent.be/tutorial/mediation.html
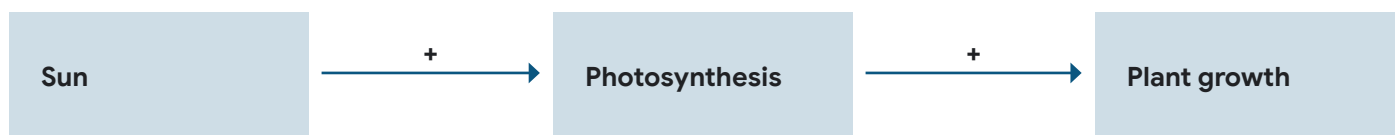
# Chapter 3's model
# Technical capabilities predict performance

Technical capabilities and processes have a positive impact on well-being and key outcomes. The effect of technical capabilities on key outcomes is partially mediated by continuous delivery, such that these technical capabilities create an environment of continuous delivery that has a downstream impact on these key outcomes.

**Well-being**

- More job satisfaction
- Less burnout
- More productivity

+

**Technical capabilities and processes**

- Trunk-based development
- Loosely coupled architecture
- Code review speed
- Continuous integration
- AI

**Key outcomes**

- Team performance
- Organizational performance
- Software delivery performance
- Operational performance

+

+

+

**Continuous delivery**
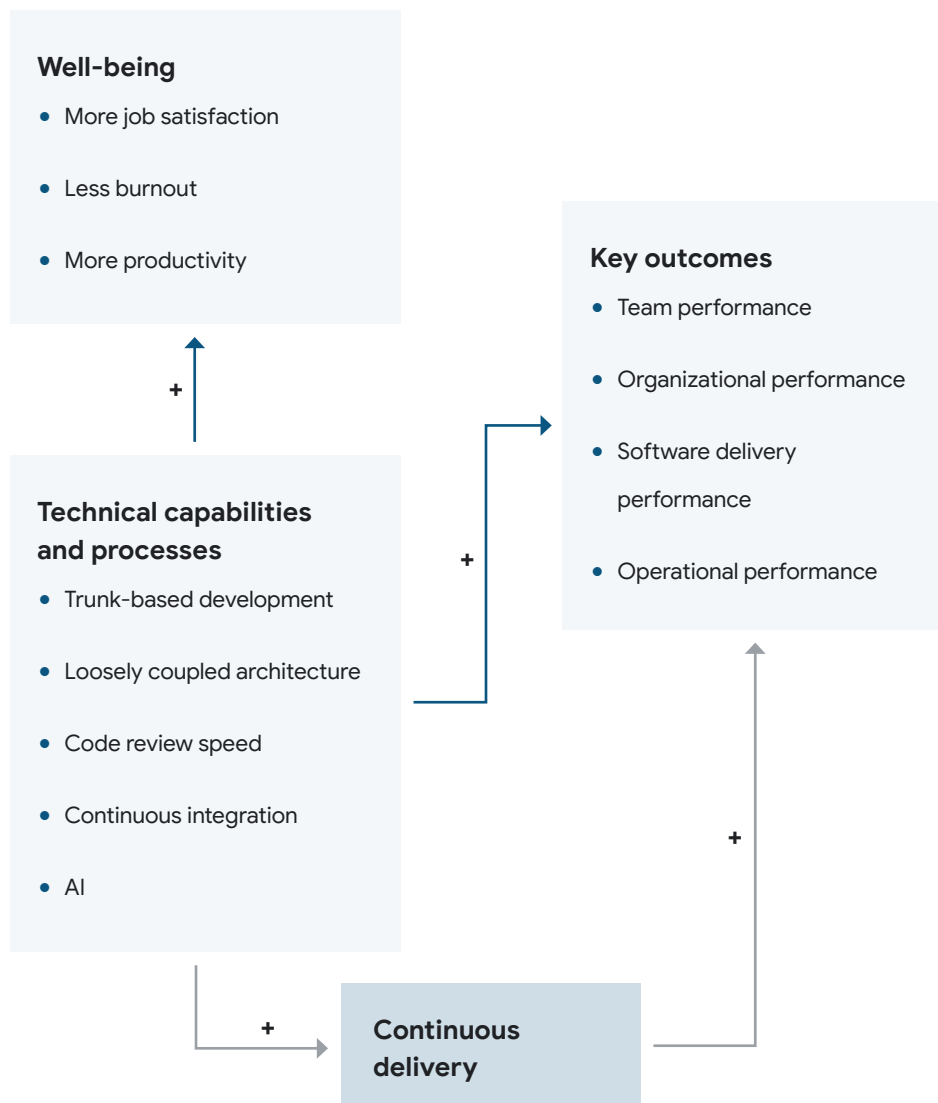
# Chapter 4's model
# Documentation is foundational

In brief, high-quality documentation has positive effects on well-being and key outcomes. This is true in part because it creates an environment where knowledge sharing is possible. High-quality documentation also helps teams establish technical capabilities and processes. Further, it helps technical capabilities and processes have an even stronger impact on key outcomes. Lastly, documentation quality doesn't lead to better well-being for everyone. We recommend reading the section for the breakdown of this complex finding.

**Well-being**

- More job satisfaction
- Less burnout
- Productivity

**Key outcomes**

- Team performance
- Organizational performance
- Software delivery performance
- Operational performance

**Gender** — Reverses

**Underrepresented** — Reverses

**Work distribution**

Attenuates

Attenuates

**Knowledge sharing**

**Documentation quality**

Amplifies

**Technical capabilities and processes**

- Trunk-based development
- Loosely coupled architecture
- Code review speed
- Continuous integration
- AI
- Continuous delivery

+

# Chapter 5's model
## Reliability unlocks performance

Here we explore the central role of operational performance. Not only does it improve well-being and key outcomes, but it also amplifies the effect of software delivery performance. Reliability practices have a nonlinear relationship with operational performance. We recommend consulting this chapter to understand those details and more.
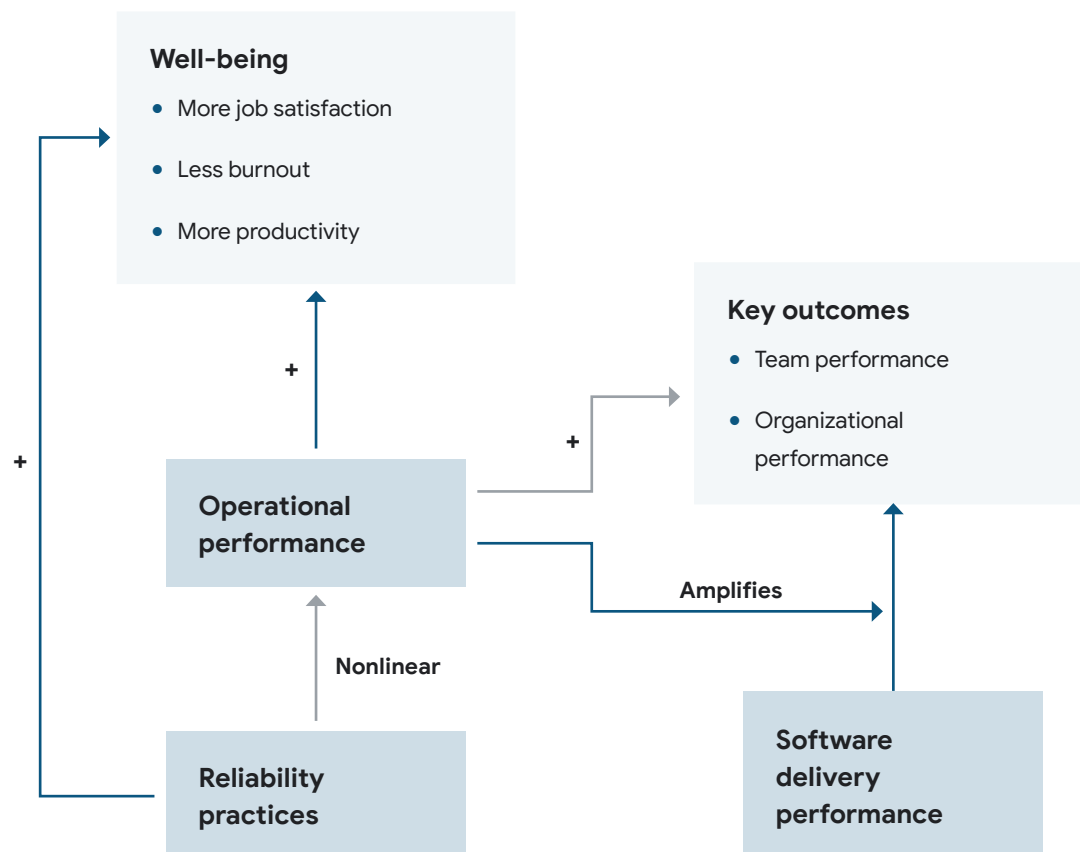
# Chapter 6's model
# Flexible infrastructure is key to success

Cloud computing has impacts on key outcomes because it provides a more
flexible infrastructure. Cloud computing also leads to better well-being.

**Flexible infrastructure**

+

+

**Cloud computing**

- Private
- Public
- Hybrid
- Multi

**Key outcomes**

- Team performance
- Organizational performance
- Software delivery performance
- Operational performance

+

**Well-being**

- More job satisfaction
- Less burnout
- More productivity
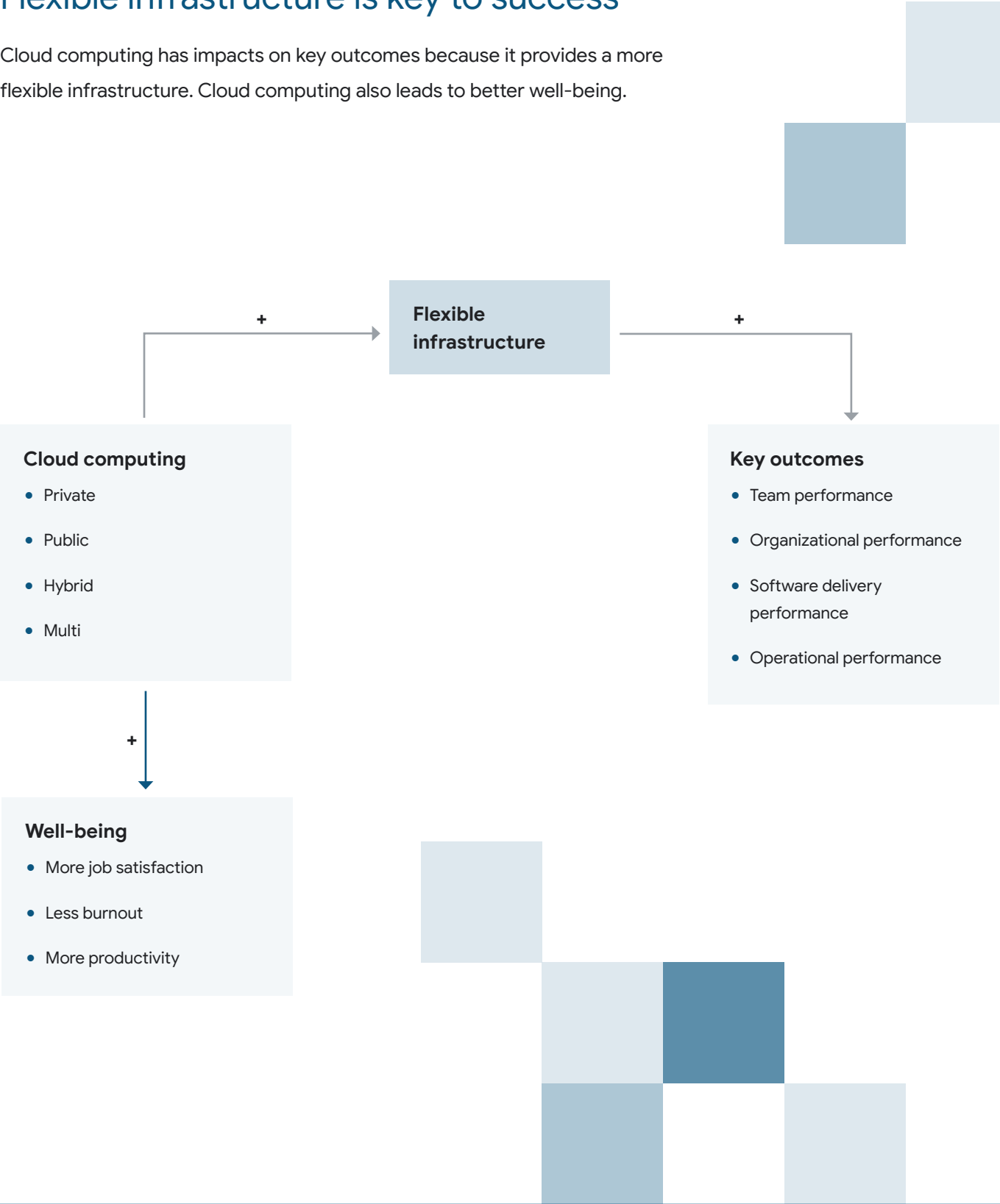
# Chapter 7's model
# None of this works without investing in culture

We can see that culture is at the center of so much in this diagram. We find that culture has a positive relationship with technical capabilities, key outcomes, and well-being.

**Technical capabilities and processes**

- Trunk-based development
- Loosely coupled architecture
- Reliability practices
- Continuous integration
- Continuous delivery

← + →

**Culture**

- Westrum generative culture
- Organizational stability
- Job security
- Flexibility
- Knowledge sharing
- User-centrisim
- Work distribution

→ +

**Key outcomes**

- Team performance
- Organizational performance
- Software delivery performance
- Operational performance

+

**Well-being**

- More job satisfaction
- Less burnout
- Productivity

# Chapter 8's models
## How, when, and why who you are matters

There are two models in this section. One explores why and when people who identify as underrepresented, and people who do not identify as men, experience higher levels of burnout. The other model explores whether documentation quality, work location, or AI can help new hires be more productive.

**Burnout**

**Toil**

**Woman or self-described gender**

**Underrepresented**

**Work distribution**

+

+

+

+

+

Attenuates

Attenuates

**New hire**  —  -  →  **Productivity**

**Documentation quality**

**Work location**

**AI**

Attenuates

Attenuates

Attenuates
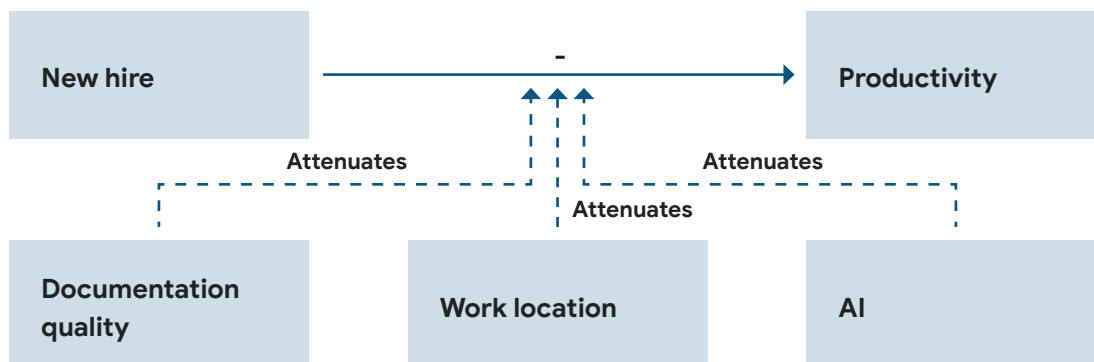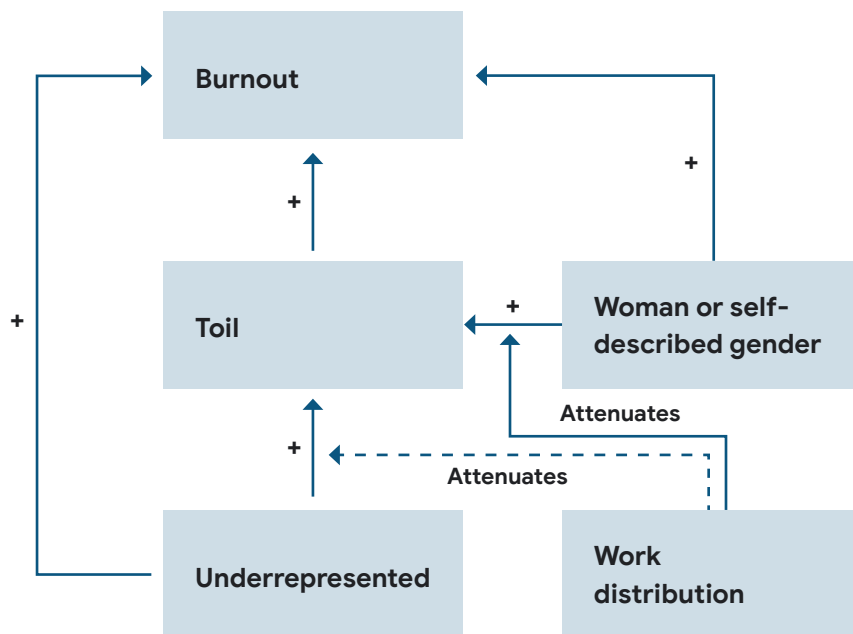
# Further reading

Join the DORA Community to discuss, learn, and collaborate on improving software delivery and operations performance. DORA.community

Take the DORA DevOps Quick Check
https://dora.dev/quickcheck

Explore the technical, process, and cultural capabilities that drive higher software delivery and organizational performance. https://dora.dev/devops-capabilities/

Find resources on SRE
https://sre.google
https://goo.gle/enterprise-roadmap-sre

Read the book: *Accelerate: The science behind devops: Building and scaling high performing technology organizations*. IT Revolution.
https://itrevolution.com/product/accelerate/

Discover an appropriate constellation of metrics for your team using the SPACE Framework.
"The SPACE of Developer Productivity: There's more to it than you think."
https://queue.acm.org/detail.cfm?id=3454124

There have been several research studies on modern code reviews. Here are a few reports to explore:

- "Expectations, Outcomes, and Challenges of Modern Code Review"
  https://dl.acm.org/doi/10.5555/2486788.2486882

- "Code Reviews - From bottlenecks to Superpowers"
  https://learning.acm.org/techtalks/codereviews

- "Modern Code Review- A Case Study at Google"
  https://dl.acm.org/doi/10.1145/3183519.3183525

- "The Choice of Code Review Process: A Survey on the State of the Practice" https://link.springer.com/chapter/10.1007/978-3-319-69926-4_9

- "Investigating the effectiveness of peer code review in distributed software development based on objective and subjective data"
  https://jserd.springeropen.com/articles/10.1186/s40411-018-0058-0

Read the book: *The No Club: Putting a Stop to Women's Dead-End Work*. Simon & Schuster.
https://www.simonandschuster.com/books/The-No-Club/Linda-Babcock/9781982152338

Publications from DORA's research program, including prior Accelerate State of DevOps Reports.
https://dora.dev/publications/

Frequently asked questions about the research and the reports. http://dora.dev/faq

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

# Appendix

## Refining how we measure software delivery performance

This year we made changes to the way we assess change failures and recovering from failure.

We changed how respondents reported their **change failure rate**. In previous years, respondents were presented with six options (0-15%, 16-30%, etc.). This year we presented respondents with a slider so that they could select any value between 0% and 100%.

We made this change for two reasons:

- Change failure rate has always behaved a little differently than the other three measures of software delivery performance.[1] We theorized that this might be due in part to the size of the buckets. Providing more precision in the answer might yield better statistical performance. We were able to validate this theory.

- We have a hypothesis that teams have a better understanding of their change failure rate today than they did when the research began almost a decade ago. We do not really have a way to validate this hypothesis.

We asked about recovering from failures two different ways this year:

- Previously, we asked this question: "For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (for example, unplanned outage, service impairment)?"

- This year, we added the following qualifiers to the question (differences are in bold here but were not bold in the survey): "For the primary application or service you work on, how long does it generally take to restore service **after a change to production or release to users results in degraded service (for example, lead to service impairment or service outage) and subsequently require remediation (for example, require a hotfix, rollback, fix forward, or patch)**?"

---

[1] Forsgren, N., Humble, J., and Kim, G. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations* (IT Revolution Press, 2018), 37–38.

The previous way of asking about recovery times did not allow for a distinction between a failure initiated by a software change and a failure initiated by something like an earthquake interrupting service at a data center. We had a hypothesis that the more precise language would allow us to compare similar failure types to one another, and that the language would be more statistically aligned with the other three measures of software delivery performance.

We are now using the term "Failed deployment recovery time" to distinguish our measure from the more generic "time-to-restore" that we've used in the past and sometimes abbreviated to "MTTR." MTTR has caused some confusion in the community: is that "M" for mean or median? Additionally, practitioners seeking to learn more from failures, such as those in the resilience engineering space, are moving past MTTR as a reliable measure for guiding learning and improvement.[2]

The newly added question and a new metric, failed deployment recovery time, are more in line with the spirit of measuring software delivery performance.

[2] "Moving Past Simple Incident Metrics: Courtney Nash on the VOID"
https://www.infoq.com/articles/incident-metrics-void/

# The math behind the comparisons

Throughout this report, there are stats that indicate that having a higher measurement for a given variable leads to N times higher scores on something else. What is this? What is it relative to? Here is our recipe for creating these formulas:

1. Create a formula from the model evaluated using regression techniques that account for potential biasing pathways:

   > Happiness ~ 5.64 + 0.19 *sunshine + 0.14 *temperature

2. For the variable of interest, say sunshine, find high and low values to compare:

   > mean = 6.3; sd = 1.4
   >
   > High sunshine = 1 sd above mean = mean + sd = 6.3 + 1.4 = 7.7
   >
   > Low sunshine = 1 sd below mean = mean - sd = 6.3 - 1.4 = 4.9

3. Calculate the mean for covariates. That would be temperature in this example, which equals 0.24 (standardized temperature).

4. Fill in the formula from step one for high sunshine and low sunshine. You'll notice that only one number is different in the formula. This is how we hold all else equal and just isolate the one difference of interest.

> Predicted high sunshine happiness = 5.64 + 0.19 *7.7 + 0.14 * 0.24 = 7.1
>
> Predicted low sunshine happiness = 5.64 + 0.19 * 4.9 + 0.14 * 0.24 = 6.6

5. Calculate the ratio:

$$\frac{\text{predicted high sunshine happiness}}{\text{predicted low sunshine happiness}} = \frac{7.1}{6.6} = 1.1x$$
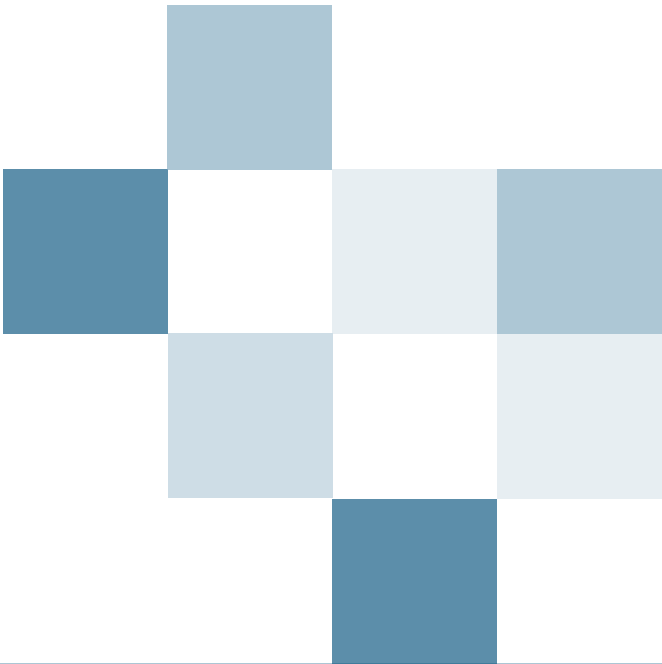
6. This ratio suggests that high levels of sunshine lead to 10% higher levels of happiness relative to low levels of sunshine.

# What is a "simulation"?

It isn't that we made up the data. We use Bayesian statistics to calculate a **posterior**, which tries to capture "the expected frequency that different parameter values will appear."[3] The "simulation" part is drawing from this posterior more than 1,000 times to explore the values that are most credible for a parameter (mean, beta weight, sigma, intercept, etc.) given our data. "Imagine the posterior is a bucket full of parameter values, numbers such as 0.1,0.7,0.5, 1, etc. Within the bucket, each value exists in proportion to its posterior probability, such that values near the peak are much more common than those in the tails."[4]

This all amounts to our using simulations to explore possible interpretations of the data and get a sense of how much uncertainty there is. You can think of each simulation as a little AI that knows nothing besides our data and a few rules trying to fill in a blank (parameter) with an informed guess. You do this 4,000 times and you get the guesses of 4,000 little AIs for a given parameter. You can learn a lot from these guesses. You can learn what the average guess is, between which values do 89% of these guesses fall, how many guesses are above a certain level, how much variation is there in these guesses, etc. You can even do fun things like combine guesses (simulations) across many models.

When we show a graph with a bunch of lines or a distribution of potential values, we are trying to show you what is most plausible given our data and how much uncertainty there is.

[3] McElreath, Richard. Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC, 2018, pg. 50
[4] McElreath, Richard. Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC, 2018, pg. 52