ACCELERATE

State of DevOps

2019

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The Accelerate State of DevOps Report represents six years of research and data from over 31,000 professionals worldwide. It is the largest and longest-running research of its kind, providing an independent view into the practices and capabilities that drive high performance. The results let us understand the practices that lead to excellence in technology delivery and powerful business outcomes.

Our research employs rigorous statistical methods to present data-driven insights about the most effective and efficient ways to develop and deliver technology. Cluster analysis allows teams to benchmark against the industry, identifying themselves as low, medium, high, or elite performers at a glance.

Teams can then leverage the findings of our predictive analysis to identify the specific capabilities they can use to improve their software delivery performance and ultimately become an elite performer.

This year, we also investigate the ways in which organizations can support engineering productivity through

initiatives such as supporting information search, more usable deployment toolchains, and reducing technical debt through flexible architecture, code maintainability, and viewable systems.

Our research continues to show that the industry-standard Four Key Metrics[1] of software development and delivery drive organizational performance in technology transformations. This year's report revalidates previous findings that it is possible to optimize for stability without sacrificing speed.

We also the identify the capabilities that drive improvement in the Four Key Metrics, including technical practices, cloud adoption, organizational practices (including change approval processes), and culture.

For organizations seeking guidance on how to improve, we point to the only real path forward: Start with foundations, and then adopt a continuous improvement mindset by identifying your unique constraint (or set of constraints). Once those constraints no longer hold you back, repeat the process. We also provide guidance on the most effective strategies for enacting these changes.

1 https://www.thoughtworks.com/radar/techniques/four-key-metrics

KEY FINDINGS

1 **The industry continues to improve, particularly among the elite performers.**

The proportion of our highest performers has almost tripled, now comprising 20% of all teams. This shows that excellence is possible—those that execute on key capabilities see the benefits.

2 **Delivering software quickly, reliably, and safely is at the heart of technology transformation and organizational performance.**

We see continued evidence that software speed, stability, and availability contribute to organizational performance (including profitability, productivity, and customer satisfaction). Our highest performers are twice as likely to meet or exceed their organizational performance goals.

3

The best strategies for scaling DevOps in organizations focus on structural solutions that build community.

High performers favor strategies that create community structures at both low and high levels in the organization, including Communities of Practice and supported Proofs of Concept, likely making them more sustainable and resilient to reorgs and product changes.

4

Cloud continues to be a differentiator for elite performers and drives high performance.

The use of cloud—as defined by NIST Special Publication 800-145—is predictive of software delivery performance and availability. The highest performing teams were 24 times more likely than low performers to execute on all five capabilities of cloud computing.

5

Productivity can drive improvements in work/life balance and reductions in burnout, and organizations can make smart investments to support it.

To support productivity, organizations can foster a culture of psychological safety and make smart investments in tooling, information search, and reducing technical debt through flexible, extensible, and viewable systems.

6

There's a right way to handle the change approval process, and it leads to improvements in speed and stability and reductions in burnout.

Heavyweight change approval processes, such as change approval boards, negatively impact speed and stability. In contrast, having a clearly understood process for changes drives speed and stability, as well as reductions in burnout.

WHO TOOK THE SURVEY?

DORA's research provides insight into software development and DevOps practices applied in industry, backed by scientific studies spanning six years with over 31,000 survey responses from working professionals. This year, almost 1,000[2] individuals from a range of industries around the world added their voices to the 2019 Report. Overall, we see similar representation across key demographic and firmographic measures when compared to last year, other than a noticeable drop in the reported percentage of women on teams.

2 With almost 1,000 respondents, our analyses have a 3% margin of error assuming 23 million software professionals worldwide and a 95% confidence interval.
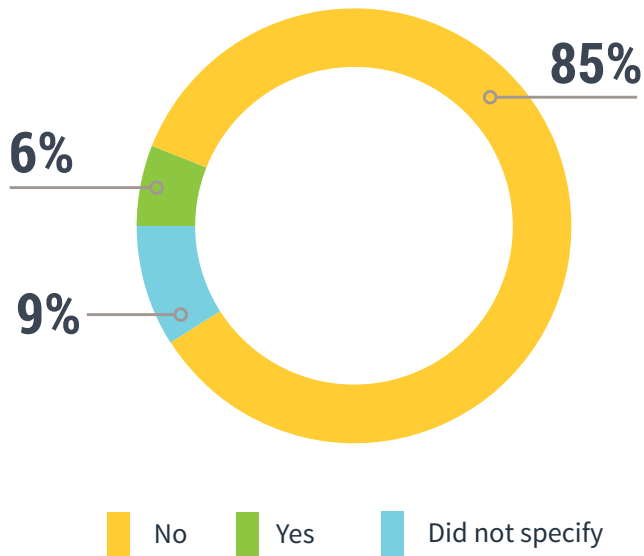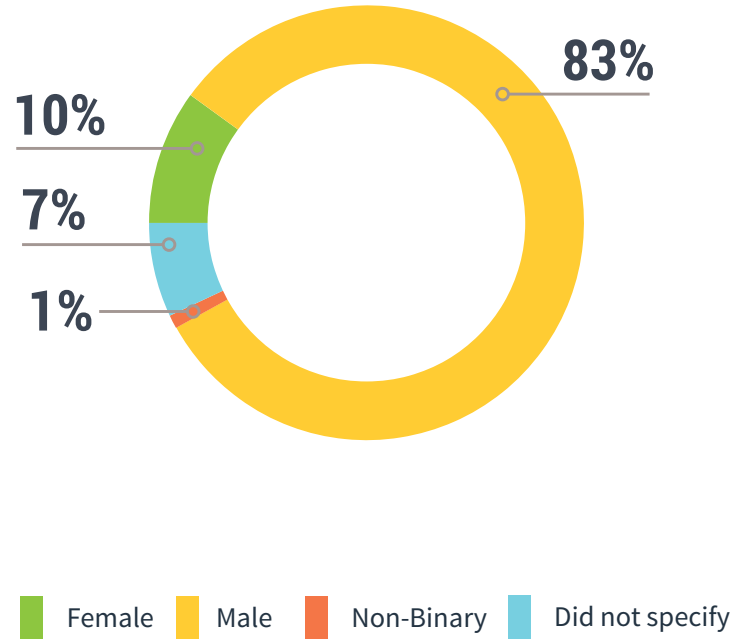
DEMOGRAPHICS & FIRMOGRAPHICS

Compared to last year, we see consistent representation of respondents across key demographic categories that include gender, disability, and underrepresented groups. While we see similar gender makeup among our survey respondents overall, the reported percentage of women on teams fell compared to last year.

We also saw consistent representation across key firmographic categories including company size, industry, and region. The majority of respondents work as engineers or managers within the technology industry. We continue to have diverse representation across departments from consultants, coaches, and sales/marketing roles. Additionally, we continue to see industry representation from highly regulated organizations in financial services, government, healthcare, and retail companies.

GENDER

Gender breakouts from this year's survey responses remain consistent with 83% male in 2019 (vs. 83% last year), 10% female (vs 12% last year), and <1% non-binary (vs <1% last year).[3]

Respondents this year stated that only 16% of teams include women (median), representing a dip from 25% reported last year.

83%
10%
7%
1%

■ Female ■ Male ■ Non-Binary ■ Did not specify

85%
6%
9%

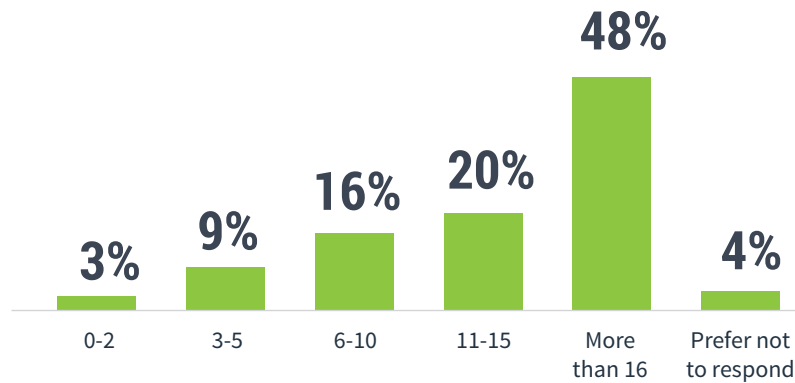■ No ■ Yes ■ Did not specify

DISABILITY

Disability is identified along six dimensions that follow guidance from the Washington Group Short Set.
This is the second year we have asked about disability and it has stayed consistent at 6% in 2018 and 2019.[4]

DEMOGRAPHICS

3 This is similar to proportions reported by the Stack Overflow Developer Survey 2019, which includes 90% men and 10% women. They do not include non-binary and "did not specify." https://insights.stackoverflow.com/survey/2019

4 This is consistent with proportions seen elsewhere in industry; e.g., the Stack Overflow Developer Survey 2019, which reports 6% of total respondents identify as having a disability. https://insights.stackoverflow.com/survey/2019

UNDERREPRESENTED GROUPS

Identifying as a member of an underrepresented group can refer to race, gender, or another characteristic. This is the third year we have captured this data and it has stayed relatively consistent from 13% in 2018 to 14% in 2019.*

76%

14%

11%

- No
- Yes
- Did not specify

* > 100% due to rounding

YEARS OF EXPERIENCE

Similar to last year, a large portion of respondents have more than 16 years of experience (50% last year), followed by 11-15 years of experience (also 20% last year).

3% 0-2
9% 3-5
16% 6-10
20% 11-15
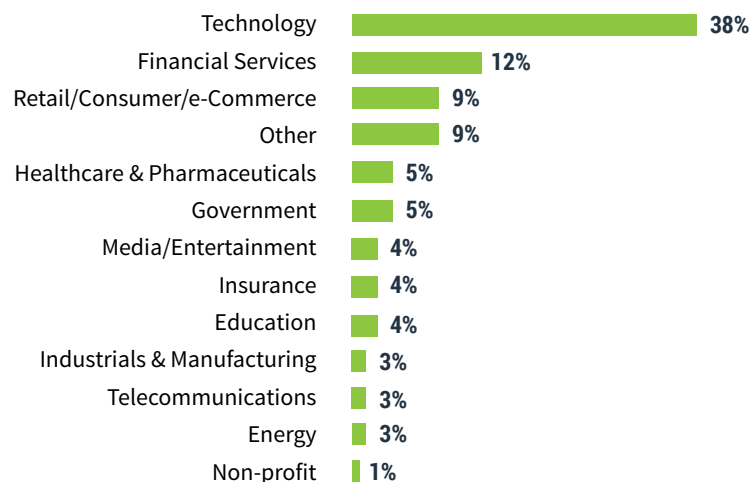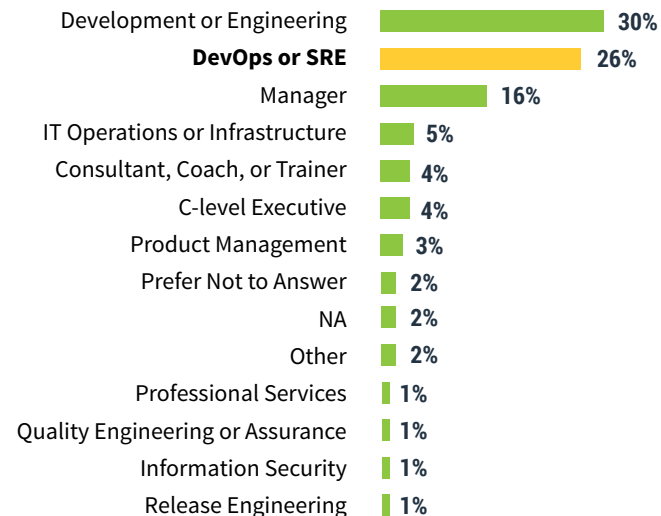**48%** More than 16
4% Prefer not to respond

Overall demographic breakouts in 2019 remain consistent with 2018, with slight percentage variances year to year that fall within the margin of error.
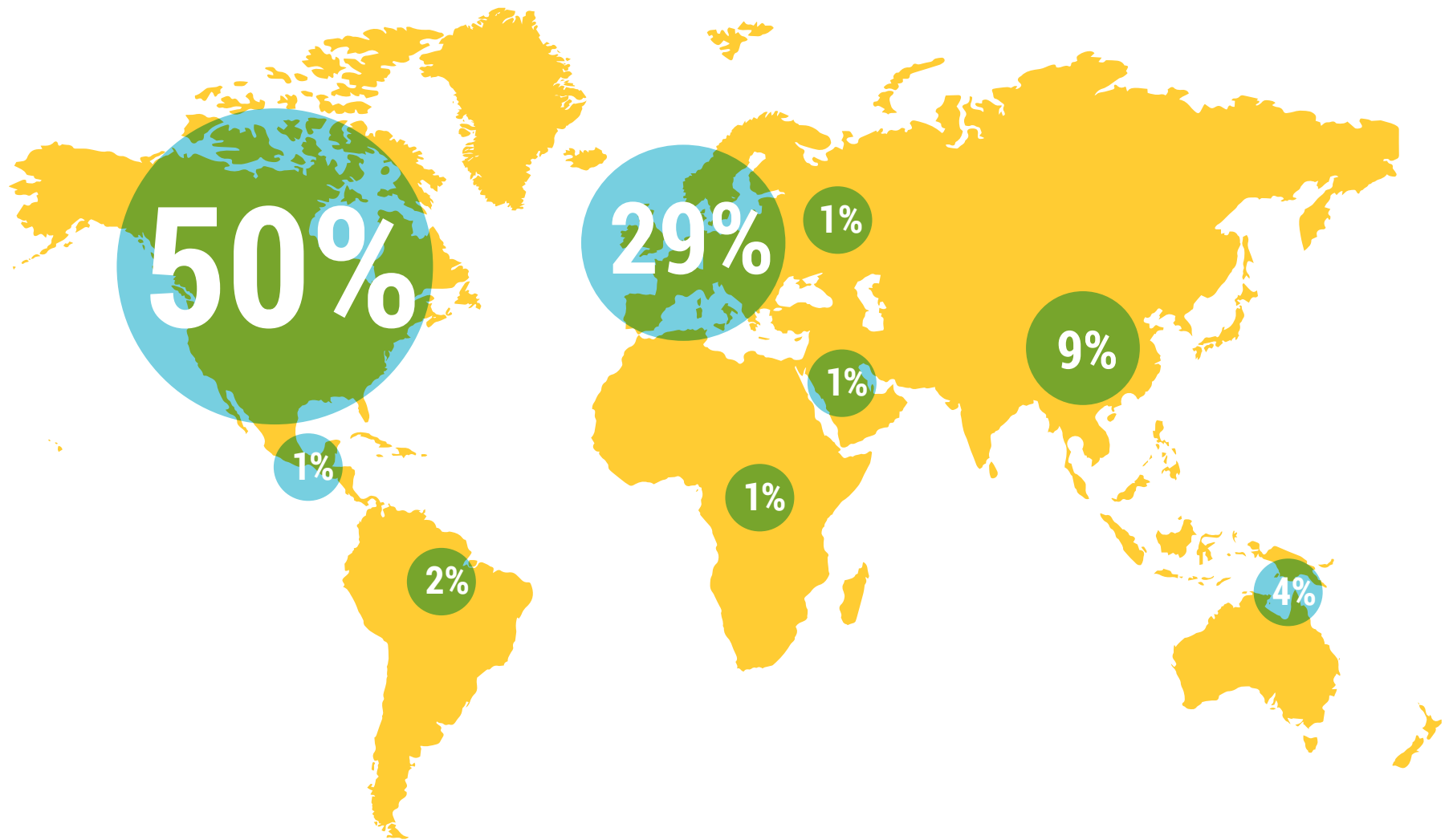
DEMOGRAPHICS

DEPARTMENTS

Participants who work in DevOps teams have increased since we began our study, reporting 16% in 2014, 19% in 2015, 22% in 2016, and holding steady around 27% for the past three years. (Note this is within the margin of error.)

Department	%
Development or Engineering	30%
DevOps or SRE	26%
Manager	16%
IT Operations or Infrastructure	5%
Consultant, Coach, or Trainer	4%
C-level Executive	4%
Product Management	3%
Prefer Not to Answer	2%
NA	2%
Other	2%
Professional Services	1%
Quality Engineering or Assurance	1%
Information Security	1%
Release Engineering	1%

FIRMOGRAPHICS

Industry	%
Technology	38%
Financial Services	12%
Retail/Consumer/e-Commerce	9%
Other	9%
Healthcare & Pharmaceuticals	5%
Government	5%
Media/Entertainment	4%
Insurance	4%
Education	4%
Industrials & Manufacturing	3%
Telecommunications	3%
Energy	3%
Non-profit	1%

INDUSTRY

Similar to last year, most respondents work within the technology industry, followed by financial services, retail, and other.

50%

29%

1%

1%

9%

1%
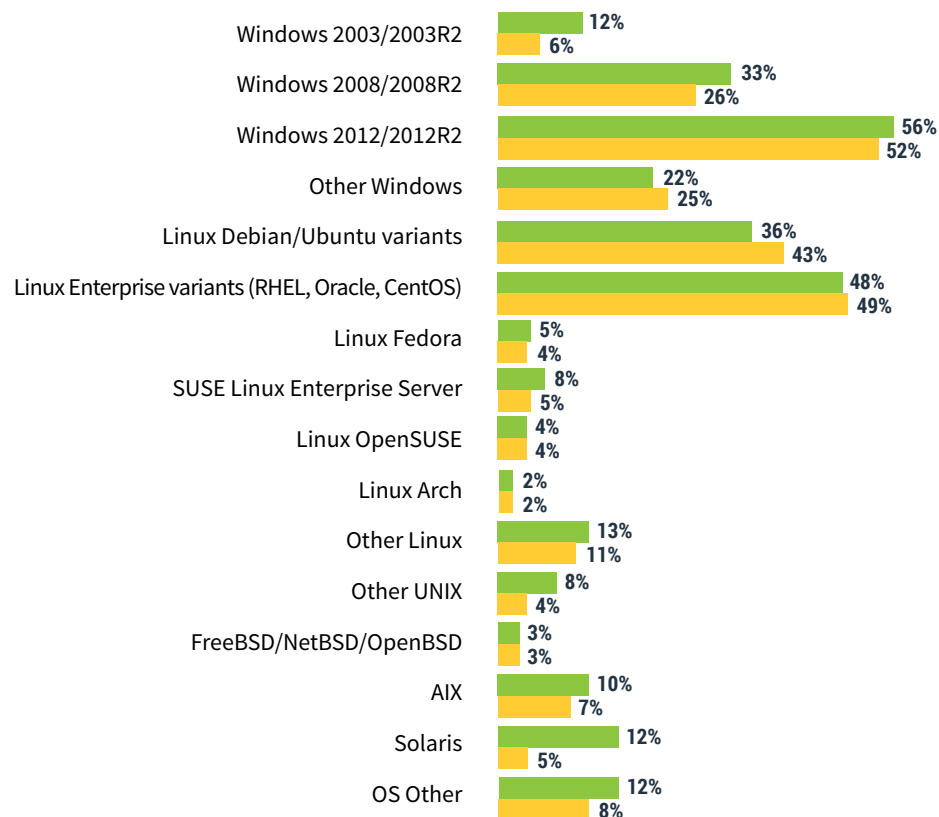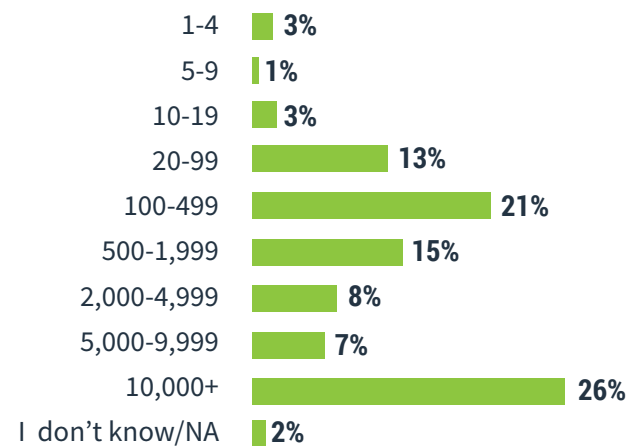
1%

2%

4%

FIRMOGRAPHICS

REGION

Consistent with last year, North America accounts for roughly half of all respondents , followed by EU/ UK at 29%. We see a drop in responses from Asia, falling from 18% last year to 9% this year.

EMPLOYEES

One out of four respondents work at very large companies (10,000+) employees, accounting for 26% of all responses, and another two out of four respondents work at companies ranging between 20-1,999 employees. These distributions are similar to the 2018 Report, though there was a drop in responses from employees working in companies with 500-1,999 employees (down 12% vs 2018) and more responses from people working in company sizes of 100-499 employees (up 7% vs 2018).

Company Size	Percentage
1-4	3%
5-9	1%
10-19	3%
20-99	13%
100-499	21%
500-1,999	15%
2,000-4,999	8%
5,000-9,999	7%
10,000+	26%
I don't know/NA	2%

FIRMOGRAPHICS

OPERATING SYSTEMS

The distribution of operating systems was fairly consistent compared to last year as well.

■ 2018
■ 2019

Operating System	2018	2019
Windows 2003/2003R2	12%	6%
Windows 2008/2008R2	33%	26%
Windows 2012/2012R2	56%	52%
Other Windows	22%	25%
Linux Debian/Ubuntu variants	36%	43%
Linux Enterprise variants (RHEL, Oracle, CentOS)	48%	49%
Linux Fedora	5%	4%
SUSE Linux Enterprise Server	8%	5%
Linux OpenSUSE	4%	4%
Linux Arch	2%	2%
Other Linux	13%	11%
Other UNIX	8%	4%
FreeBSD/NetBSD/OpenBSD	3%	3%
AIX	10%	7%
Solaris	12%	5%
OS Other	12%	8%

HOW DO WE COMPARE?

This section functions as your DevOps benchmark assessment. We use rigorous statistical methods to examine how teams are developing, delivering, and operating software systems. Benchmarks for elite, high, medium, and low performers show where you are in the context of multiple important analyses throughout the report. We also identify trends year over year.
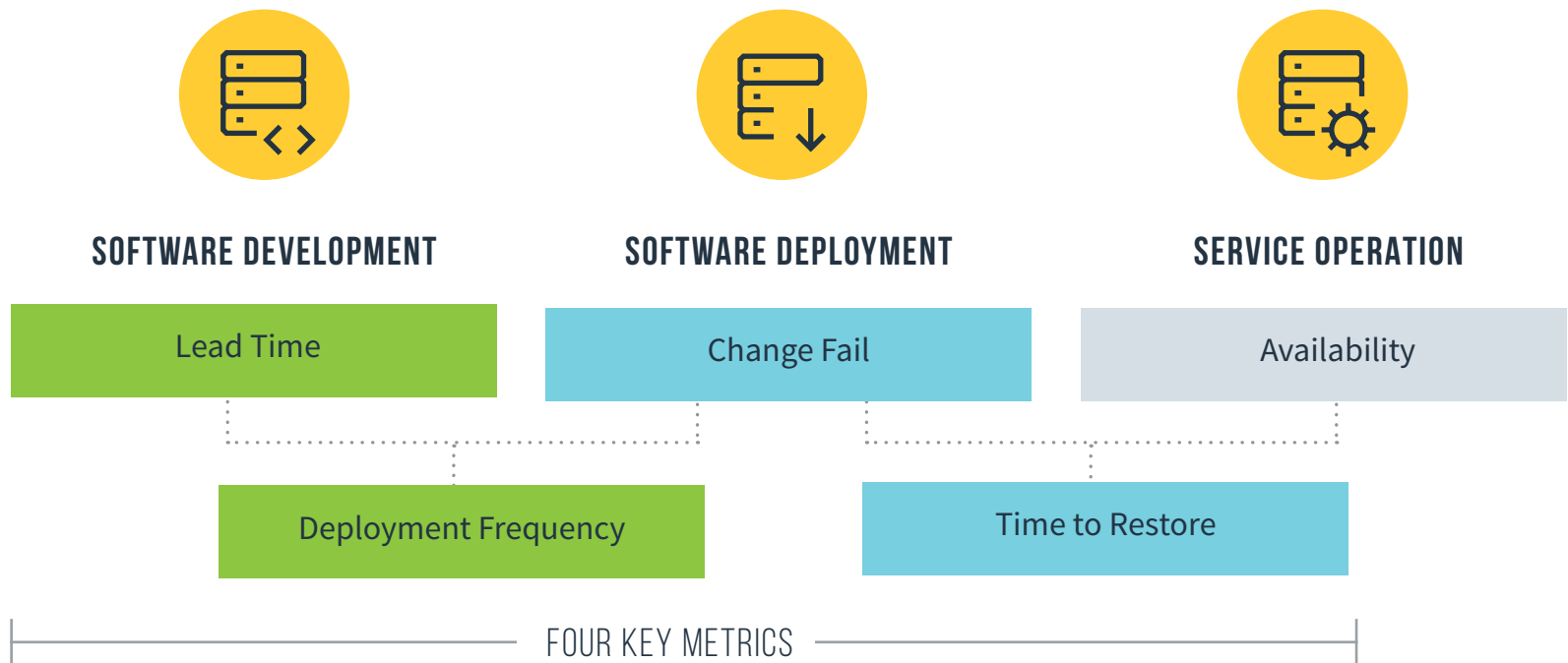
SOFTWARE DELIVERY AND OPERATIONAL PERFORMANCE

Organizations increasingly rely on their ability to deliver and operate software systems to achieve their goals. To compare performance on this key outcome metric, the industry needs a way to measure the effectiveness of their development and delivery practices. Over the last six years we have developed and validated four metrics that provide a high-level systems view of software delivery and performance and predict an organization's ability to achieve its goals. Last year, we added an additional metric focused on operational capabilities, and found that this measure helps organizations deliver superior outcomes. We call these five measures **software delivery and operational (SDO) performance**, which focus on system-level outcomes. This helps avoid the common pitfalls of software metrics, which often pit different functions against each other and result in local optimizations at the cost of overall outcomes.

The first four metrics that capture the effectiveness of the development and delivery process can be summarized in terms of throughput and stability. We measure the throughput of the software delivery process using lead time of code changes from check-in to release along with deployment frequency. Stability is measured using time to restore—the time it takes from detecting a user-impacting incident to having it remediated—and change fail rate, a measure of the quality of the release process.

PERFORMANCE METRICS

SOFTWARE DEVELOPMENT	SOFTWARE DEPLOYMENT	SERVICE OPERATION
Lead Time	Change Fail	Availability
Deployment Frequency	Time to Restore	

FOUR KEY METRICS

Many professionals approach these metrics as representing a set of trade-offs, believing that increasing throughput will negatively impact the reliability of the software delivery process and the availability of services. For six years in a row, however, our research has consistently shown that speed and stability are outcomes that enable each other. Cluster analysis of the four software delivery measures in the 2019 data reveals four distinct performance profiles, with statistically significant differences in throughput and stability measures among them.[5] As in previous years, our highest performers do significantly better on all four measures, and low performers do significantly worse in all areas.

In addition to speed and stability, availability is important for operational performance. At a high level, availability represents an ability for technology teams and organizations to keep promises and assertions about the software they are operating. Notably, availability is about ensuring a product or service is available to and can be accessed by your end users.[6]

Availability reflects how well teams define their availability targets, track their current availability, and learn from any outages, making sure their feedback loops are complete. The items used to measure availability form a valid and reliable measurement construct.

---

5 Availability is not included in our cluster analysis because availability measures do not apply the same way for software solutions that are not provided in the form of services, such as packaged software or firmware.

6 Teams can define their availability goals using Service Level Agreements (SLAs) and Service Level Objectives (SLOs) and measure their performance using Service Level Indicators (SLIs). For more information on developing SLAs, SLOs, and SLIs, you can check out Site Reliability Engineering: How Google Runs Production Systems (2016) by Beyer et al.

Aspect of Software Delivery Performance*	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per day and once per week	Between once per week and once per month	Between once per month and once every six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one day	Between one day and one week	Between one week and one month	Between one month and six months
Time to restore service 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 (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day[a]	Less than one day[a]	Between one week and one month
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0-15%[b,c]	0-15%[b,d]	0-15%[c,d]	46-60%

Medians reported because distributions are not normal.

All differences are significantly different based on Tukey's post hoc analysis except where otherwise noted.

a,b,c Means are significantly different based on Tukey's post hoc analysis; medians do not exhibit differences because of underlying distributions.

d Means are not significantly different based on Tukey's post hoc analysis.

*For a visual presentation of the Four Metrics, please see Appendix A.

We also confirmed last year's finding that better software delivery goes hand-in-hand with higher availability. Analysis shows that availability measures are significantly correlated with software delivery performance profiles, and elite and high performers consistently reported superior availability, with elite performers being 1.7 times more likely to have strong availability practices.[7]

Industry velocity is increasing

Many analysts are reporting the industry has "crossed the chasm" with regards to DevOps and technology transformation, and our analysis this year confirms these observations. Industry velocity is increasing and speed and stability are both possible, with shifts to cloud technologies fueling this acceleration. This reaffirms the importance of technology that enables organization to deliver value to their stakeholders.

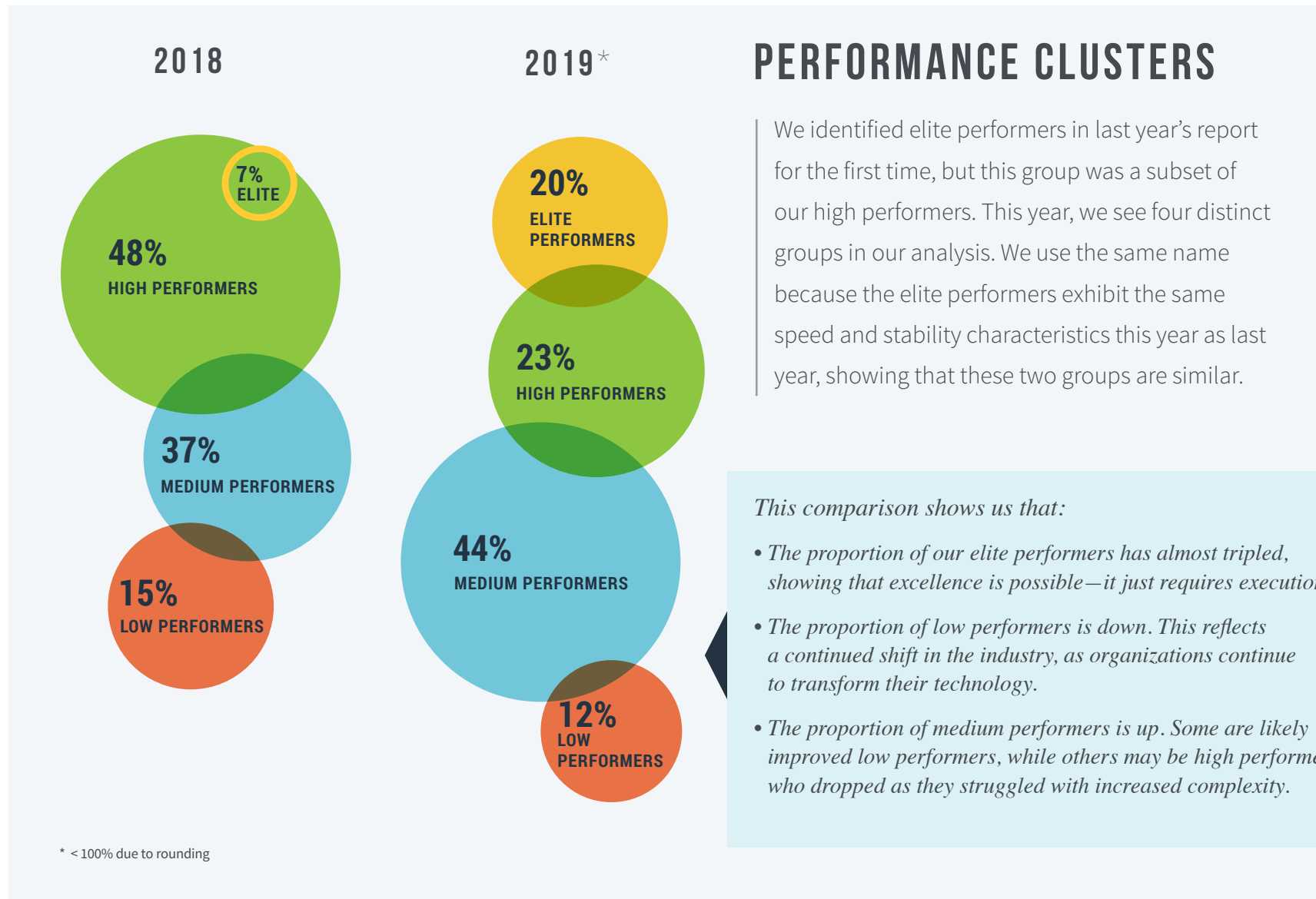7 It should also be noted that none of these practices apply solely to the cloud.

INDUSTRY AND ORGANIZATION IMPACTS ON SDO PERFORMANCE

We ran additional analyses (e.g., using control variables) to see if industry and organization size had a significant effect on SDO performance. We found no evidence that industry has an impact with the exception of retail, suggesting that organizations of all types and sizes, including highly regulated industries such as financial services and government, can achieve high levels of performance. Our results for the retail industry suggest that those in retail see some benefits in speed and stability.

We found evidence that enterprise organizations (those with more than 5,000 employees) are lower performers when compared to those with fewer than 5,000 employees. This is likely due to several factors seen in large organizations, most notably heavyweight process and controls as well as tightly coupled architectures that introduce delay and associated instability. We urge enterprises not to take these findings as an excuse to suffer poor performance, but recognize that excellence is possible, embark on a program of continuous improvement, and look to other enterprise organizations that have achieved elite performance for inspiration and guidance.

We compared the proportions of each performance cluster in 2018 and 2019:

2018

- **7% ELITE**
- **48% HIGH PERFORMERS**
- **37% MEDIUM PERFORMERS**
- **15% LOW PERFORMERS**

2019 *

- **20% ELITE PERFORMERS**
- **23% HIGH PERFORMERS**
- **44% MEDIUM PERFORMERS**
- **12% LOW PERFORMERS**

* < 100% due to rounding

PERFORMANCE CLUSTERS

We identified elite performers in last year's report for the first time, but this group was a subset of our high performers. This year, we see four distinct groups in our analysis. We use the same name because the elite performers exhibit the same speed and stability characteristics this year as last year, showing that these two groups are similar.

This comparison shows us that:

- *The proportion of our elite performers has almost tripled, showing that excellence is possible—it just requires execution.*

- *The proportion of low performers is down. This reflects a continued shift in the industry, as organizations continue to transform their technology.*

- *The proportion of medium performers is up. Some are likely improved low performers, while others may be high performers who dropped as they struggled with increased complexity.*

ELITE PERFORMERS

Comparing the elite group against the low performers, we find that elite performers have…

208 TIMES MORE
frequent code deployments

106 TIMES FASTER
lead time from commit to deploy

2,604 TIMES FASTER
time to recover from incidents

7 TIMES LOWER
change failure rate

(changes are $\frac{1}{7}$ as likely to fail)

■ Throughput   ■ Stability

THROUGHPUT

Deployment frequency

The elite group reported that it routinely deploys on-demand and performs multiple deployments per day, consistent with the last several years. By comparison, low performers reported deploying between once per month (12 per year) and once per six months (two per year), which is a decrease in performance from last year. The normalized annual deployment numbers range from 1,460 deploys per year (calculated as four deploys per day x 365 days) for the highest performers to seven deploys per year for low performers (average of 12 deploys and two deploys). Extending this analysis shows that elite performers deploy code 208 times more frequently than low performers. It's worth noting that four deploys per day is a conservative estimate when comparing against companies such as CapitalOne that report deploying up to 50 times per day for a product,[8] or companies such as Amazon, Google, and Netflix that deploy thousands of times per day (aggregated over the hundreds of services that comprise their production environments).

---

8 In 2017: https://www.informationweek.com/devops/capital-one-devops-at-its-core/d/d-id/1330515

Change lead time

Similarly, elite performers report change lead times of less than one day, with change lead time measured as the time from code committed to having that code successfully deployed in production. This is a small decrease in performance from last year, when our highest performers reported change lead times of less than one hour. In contrast to our elite performers, low performers required lead times between one month and six months. With lead times of 24 hours for elite performers (a conservative estimate at the high end of "less than one day") and 2,555 hours for low performers (the mean of 730 hours per month and 4,380 hours over six months), the elite group has 106 times faster change lead times than low performers.

STABILITY

Time to restore service

The elite group reported time to restore service of less than one hour, while low performers reported between one week and one month. For this calculation, we chose conservative time ranges: one hour for high performers and the mean of one week (168 hours) and one month (5,040 hours) for low performers. Based on these numbers, elites have 2,604 times faster time to restore service than low performers. As previously noted, time to restore service performance stayed the same for both elite and low performers when compared to the previous year.

Change failure rate

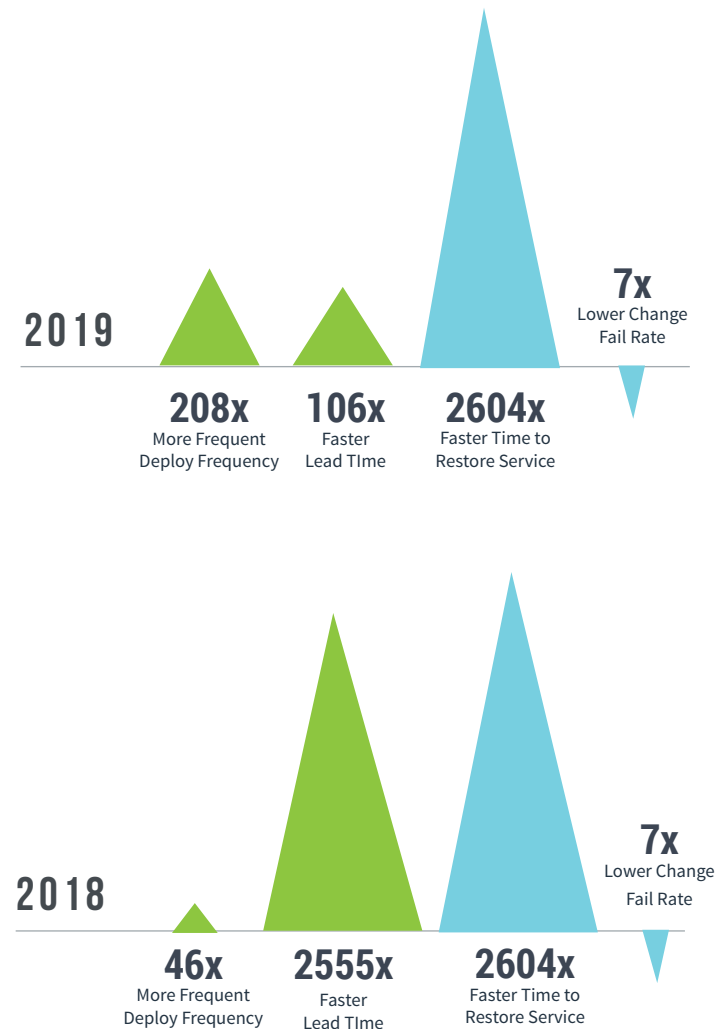Elite performers reported a change failure rate between zero and 15%, while low performers reported change failure rates of 46% to 60%. The mean between these two ranges shows a 7.5% change failure rate for elite performers and 53% for low performers. This represents change failure rates for elite performers that are seven times better than low performers. As noted earlier, change failure rates stayed the same for both elite and low performers when compared to the previous year.

All of the measures shown are relative; that is, they compare the highest and the lowest performers each year. From 2018 to 2019, the gap for all performance metrics between the lowest and highest performers increased or stayed the same, with the exception of lead time for changes. The increased gap in deploy frequency indicates a decrease in performance among low performers, which may be due to growing complexity in environments and therefore difficulty in delivering software. The reduced ratio of the lowest to the highest performers in lead time represents a reduction in the performance of the highest performing group, which is seeing lead times increase from less than an hour to between an hour and a day. This may reflect the trend in more heavyweight code review and approval processes that have become popular in recent years.

SOFTWARE DELIVERY PERFORMANCE

Comparing highest to lowest performers.

2019

208x
More Frequent
Deploy Frequency

106x
Faster
Lead TIme

2604x
Faster Time to
Restore Service

7x
Lower Change
Fail Rate

2018

46x
More Frequent
Deploy Frequency

2555x
Faster
Lead TIme

2604x
Faster Time to
Restore Service

7x
Lower Change
Fail Rate

HOW TO USE THE RESEARCH MODELS

This year's Report is designed to help drive improvement in both performance and productivity using two research models. You may wonder, why are there two research models? How are they different? How are they similar? And most importantly, how can I use them to help me make decisions and guide my own work?

Start by identifying your goal...

SDO & ORGANIZATIONAL PERFORMANCE	◀◀◀ START ▶▶▶	PRODUCTIVITY

CULTURE OF PSYCHOLOGICAL SAFETY

CHANGE MANAGEMENT

USEFUL, EASY-TO-USE TOOLS

CLOUD

INTERNAL SEARCH

TECHNICAL PRACTICES

EXTERNAL SEARCH

DISASTER RECOVERY TESTING

TECHNICAL DEBT

If you want to improve SDO performance or organizational performance, look at the model with those constructs, and <u>head to that section of the report</u> for guidance on which capabilities you should focus on.

If you want to improve productivity, look at the model with productivity as a construct, and <u>head to that section of the report</u> for guidance on which capabilities you should focus on.

How to use the models to guide your transformation

- Identify the capabilities that will improve your goal (that is, those with arrows that point to the construct you want to improve). As we've identified in this report, these are your candidate capabilities for improvement. (For SDO and organizational performance, we have also identified additional capabilities in our previous five years of research.)[9]

- Remember to accelerate your transformation by starting with a solid foundation and then focusing on the capabilities that are constraints: What capabilities cause the biggest delays? What are the biggest headaches? Where are the biggest problems? Pick three to five and dedicate resources to solving these first. Don't worry if you still have problems; by focusing on the biggest problems now, you remove bottlenecks, discover synergies, and avoid unnecessary work.

- There are other important outcomes from this work. Benefits from pursuing improvements in SDO and organizational performance include reducing burnout and deployment pain (which we researched in 2016 and 2017), and improving security outcomes (which we researched in 2017 and 2018), and culture (researched in years 2014 through 2019). Additional benefits from improving productivity include improving work/life balance and reducing burnout.

How to read the research models

We use a structural equation model (SEM), which is a predictive model used to test relationships. Each box represents a construct we measured in our research, and each arrow represents relationships between the constructs. A larger box that contains boxes (constructs) is a second-order construct. A light blue box with a dotted line to another construct indicates a control variable. (See pages <u>31</u> and <u>57</u> for full models.) Constructs in bold represent those that we investigate for the first time this year. Constructs with a dark bold outline are common team and organizational goals: SDO performance and organizational performance or productivity. Keep these in mind as you identify your goals and read the models.

To interpret the models, all lines with arrows can be read using the words "predicts," "affects," "drives," or "impacts." For example, the second-order construct SDO performance is comprised of the constructs software delivery performance and availability, and these together drive organizational performance. The construct disaster recovery testing drives availability. We indicate that disaster recovery testing is a newly investigated construct this year by marking it in bold. An arrowed line with a (-) next to it indicates a negative impact between two constructs; for example, technical debt negatively impacts (or reduces) productivity.

You may notice there's some overlap in the two research models.

This is because the goals—SDO performance and productivity—are related in many ways. The outcomes are about making and delivering technology in superior ways, and in ways that deliver value to organizations and to individuals. It makes sense that some of the things we do to support the work of software delivery will also benefit the productivity of those who develop and deliver software. Yet while they are similar, they still measure different outcomes and so we conduct our analysis separately. Thus, they are in two different research models.

What the overlap in the two research models tells us

- Making smart investments in the pursuit of SDO performance can reduce burnout, and better productivity can lead to reductions in burnout as well. This should be encouraging to organizations and technologists alike, as the demands of work continue to grow. We note that having a good work/life balance is key to reducing burnout.

- A culture of psychological safety contributes to SDO performance, organizational performance, and productivity, showing that growing and fostering a healthy culture reaps benefits for organizations and individuals.

- Investments in code maintainability, loosely coupled architecture, and monitoring help support SDO performance (via continuous delivery) and productivity (via reductions in technical debt), highlighting the importance of good tooling and systems.

HOW DO WE IMPROVE SDO & ORGANIZATIONAL PERFORMANCE?

A key goal in digital transformation is optimizing software delivery performance: leveraging technology to deliver value to customers and stakeholders. Our research provides evidence-based guidance so you can focus on the capabilities and practices that matter to accelerate your transformation. This year, we also outline implementation strategies so you can set your path forward for maximum impact.
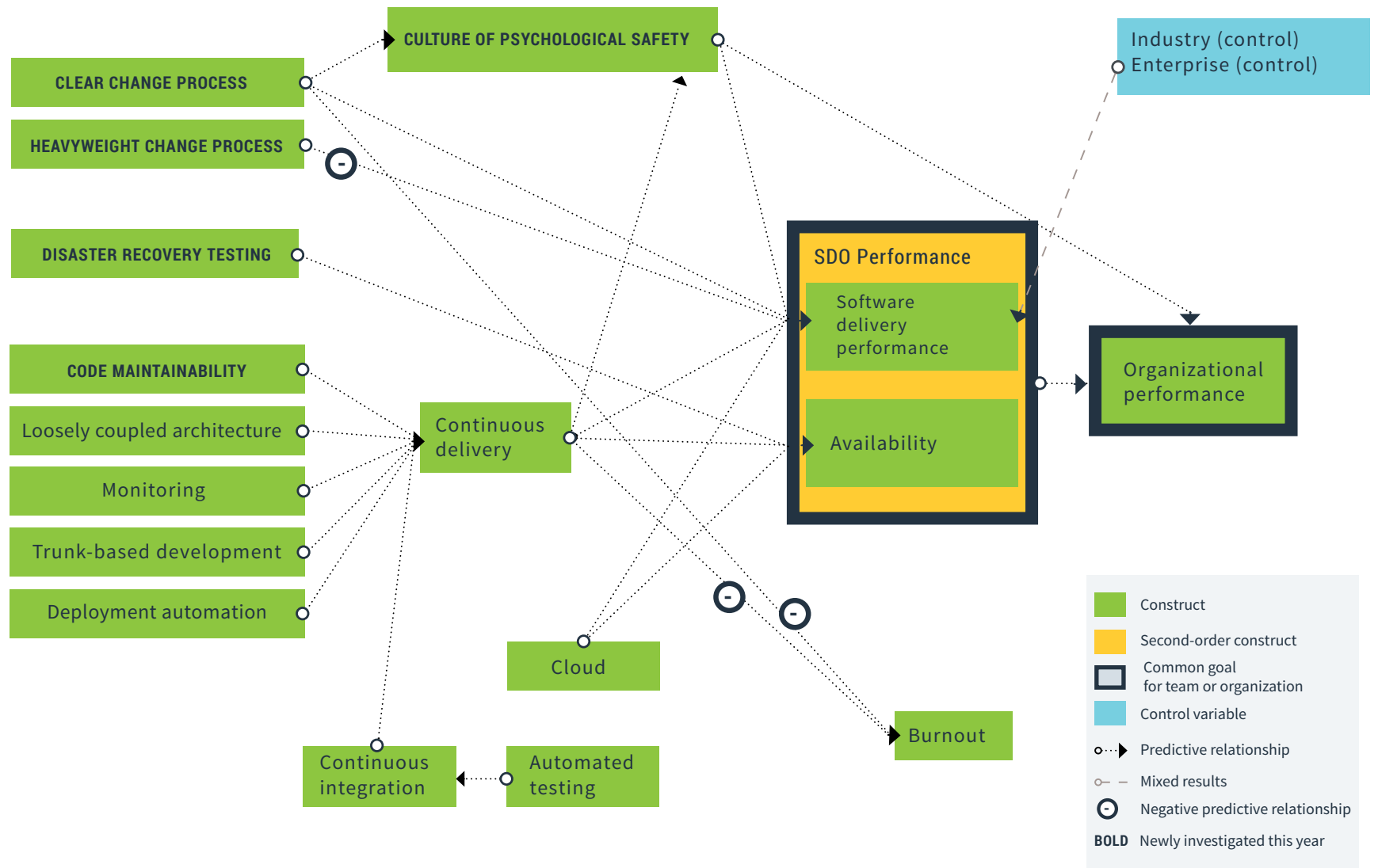
IMPROVING SDO AND ORGANIZATIONAL PERFORMANCE

Begin by focusing on the capabilities outlined in our research; they provide predictive guidelines to improve your technology delivery and deliver value. Start with foundations: Basic automation (such as version control and automated testing), monitoring, clear change approval processes, and a healthy culture. Then identify your constraints to plan your path forward. This strategy works for those just beginning transformations as well as those who have been optimizing for years. Focus resources on what is currently holding you back, then iterate: Identify constraints and choose the next target.

Use the model on page 31 to locate the goal you want to improve and identify the capabilities that impact it. For example, if your goal is to improve software delivery performance, these capabilities are culture, clear change process, continuous delivery, and cloud. Then focus on those that are your biggest constraints.

30

Accelerate: State of DevOps 2019 | How Do We Improve SDO & Organizational Performance?

SOFTWARE DELIVERY & OPERATIONAL PERFORMANCE

CULTURE OF PSYCHOLOGICAL SAFETY

CLEAR CHANGE PROCESS

HEAVYWEIGHT CHANGE PROCESS

DISASTER RECOVERY TESTING

CODE MAINTAINABILITY

Loosely coupled architecture

Monitoring

Trunk-based development

Deployment automation

Continuous delivery

Cloud

Continuous integration

Automated testing

Industry (control)
Enterprise (control)

SDO Performance
- Software delivery performance
- Availability

Organizational performance

Burnout

Legend:
- Construct
- Second-order construct
- Common goal for team or organization
- Control variable
- Predictive relationship
- Mixed results
- Negative predictive relationship
- **BOLD** Newly investigated this year

EXCEL OR DIE ›››

Our analysis found no evidence that industry made an impact in the speed and stability of software delivery, except for retail, which saw significantly better SDO performance. When we consider the crushing competitive environment of the retail industry—termed the retail apocalypse following a decade of steady closures—this should come as no surprise. Those who excel at delivering profitability, productivity, and customer satisfaction survive. Anything less than excellence leads to failure. While retailers may be at the forefront of this highly competitive shift, we see other industries such as financial services following quickly behind.

Keeping up with the rate of technological change is essential for organizations in these competitive environments who must keep demanding customers happy and satisfied while delivering consistent revenues to keep stakeholders satisfied. Retail may be the perfect example of technology delivering value, and those in other industries should learn from their experience:

- Retailers were among the first to embrace A/B testing to understand customers' buying habits, preferences, and interactions in websites and apps so they could optimize purchases. This technical ability requires more robust technical solutions and provides a powerful feedback loop to product development and marketing.

- Retailers often have some of the slimmest margins, requiring efficiency and automation to scale and respond to changes quickly.

- Retailers must be able to cope with huge swings in demand or risk going out of business—Black Friday can make or break a retailer's entire year. By leveraging the cloud, retailers can burst capacity easily and they aren't stuck having discussions about "if" or "when" they should use the cloud. They're already there.

- Retailers have figured out how to be nimble in highly regulated environments because they have to. While other industries had the luxury of blaming regulation for delayed adoption, the competitive environment forced retailers to figure out how to operate in regulated environments quickly and securely. And they're doing it. After all, you can't sell goods without processing the occasional credit card transaction.
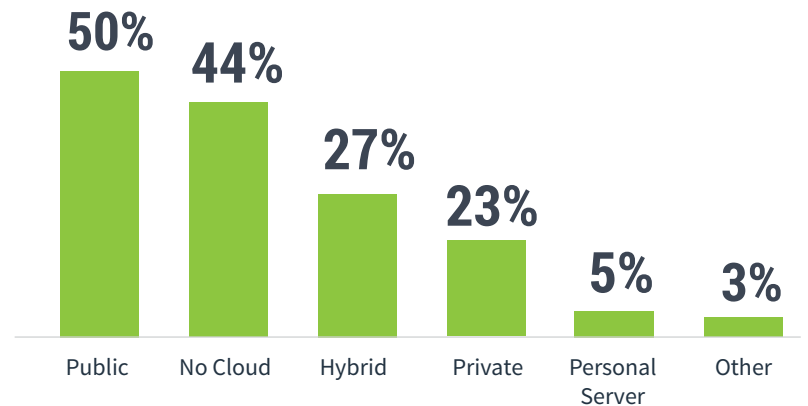
CLOUD

With the evolving nature of business, more and more organizations are choosing multi-cloud and hybrid cloud solutions. This is because these solutions offer flexibility, control, and availability in addition to performance gains.[10] In our survey, respondents indicated increased use of multi-cloud and hybrid cloud compared to last year.

We also asked respondents to indicate where their primary application work was hosted, and again saw responses that indicate there is no clear consensus on what it means to work in a hybrid or multi-cloud environment. As we stated in last year's report, hybrid is often self-defined. If respondents say they are working in a hybrid environment, then they are. This often creates frustration (and widely varying reports

among industry analysts) when experts try to discuss terms and the state of the industry: We can't compare things that we can't define and measure.

HOSTING FOR PRIMARY SERVICE OR APPLICATION

50%	44%	27%	23%	5%	3%
Public	No Cloud	Hybrid	Private	Personal Server	Other

---

10 Transform Your Business with a Hybrid and Multicloud Strategy, Tilak, March 2019.

HOW YOU IMPLEMENT CLOUD INFRASTRUCTURE MATTERS

If everyone has a different understanding of what it means to be "in the cloud," how can we actually measure its benefits? We address this limitation by focusing on **the essential characteristics of cloud computing**[*]—as defined by the National Institute of Standards and Technology (NIST)—and use that as our guide.

In our survey, 80% of respondents[11] said the primary application or service they supported was hosted on some kind of cloud platform. Using the NIST framework, we investigated the impact of essential practices on SDO performance and, for the second year in a row, found that what really matters is how teams implement their cloud services, not just that they are using a cloud technology.

---

* Terms in bold throughout this report match constructs that can be found in the research models on <u>page 31</u> and <u>57</u>.

11 Refer to the previous chart that shows where respondents' primary service or application is hosted.
Note that more than one option could be selected, so 80% of respondents selected an option that included the cloud.

Elite performers were 24 times more likely to have met all essential cloud characteristics than low performers.[12] This may explain why teams and executives who claim to have adopted cloud computing technologies also feel frustration at not reaping the promised benefits of speed and stability: Many of our survey respondents who claim to be using cloud computing haven't actually adopted the essential patterns that matter. Only 29% of respondents who said they were using cloud infrastructure agreed or strongly agreed that they met all five of the characteristics of essential cloud computing defined by NIST.

---

12 This is consistent with last year's findings, that elite performers were 23 times more likely than low performers to agree or strongly agree with all essential cloud characteristics.

FIVE ESSENTIAL CHARACTERISTICS OF CLOUD COMPUTING

% AGREED OR STRONGLY AGREED

57%

On-demand self-service
+11% from 2018

Consumers can automatically provision computing resources as needed, without human interaction from the provider.

60%

Broad network access
+14% from 2018

Capabilities can be accessed through heterogeneous platforms such as mobile phones, tablets, laptops, and workstations.

58%

Resource pooling
+15% from 2018

Provider resources are pooled in a multi-tenant model, with physical and virtual resources dynamically assigned on-demand. The customer may specify location at a higher level of abstraction such as country, state, or datacenter.

58%

Rapid elasticity
+13% from 2018

Capabilities can be elastically provisioned and released to rapidly scale outward or inward on demand, appearing to be unlimited and able to be appropriated in any quantity at any time.

62%

Measured service
+14% from 2018

Cloud systems automatically control, optimize, and report resource use based on the type of service such as storage, processing, bandwidth, and active user accounts.

Last year we found that elite performing teams were more likely to be executing on all five essential cloud characteristics, and those findings were revalidated this year. These characteristics matter when defining what it means to adopt cloud computing because they enable an actionable strategy for success: Our research shows they impact SDO performance. By focusing on execution in the cloud—whether public, private, or hybrid—any team or organization is capable of reaping the benefits of speed, stability, and availability.

SCALING THE CLOUD

A clear win from using the cloud is on-demand scaling. Teams that take advantage of dynamic scaling are able to make the infrastructure behind their service elastically react to demand from users. Teams can monitor their services and automatically scale their infrastructure as needed.

The abstractions used in the cloud have changed the ways we think about and visualize our infrastructure. For someone using a container orchestrator such as Kubernetes or Mesos, the package being shipped is a container image plus the config for deployment. Typical platform-as-service (PaaS) offerings are leaning more towards a deployment model centered around container images as the packaging method and container runtimes for execution. We see this in products such as Heroku, Google App Engine, Azure Container Instances, Cloud Foundry, and Amazon's Fargate. Serverless (also known as function-as-a-service, or FaaS)[13] has taken this one step further, simplifying deployment and allowing consumers to only worry about the execution of the application code itself and abstracting scaling, capacity planning, and maintenance away from developers and operators. Examples include AWS Lambda, Azure Functions, GCP Cloud Functions, and Zeit.

Over time, the abstractions used in the cloud have become universal standards for deployment across cloud and platform providers. Network, virtual machines, identity and access management (IAM), storage, and databases have all become the de-facto products of every cloud service provider, in addition to machine learning, Internet of Things (IoT), container solutions, language runtime solutions, and security products. We continue to see the state of products converge on the paradigms around containers, runtime languages, and application packages.

13 The term serverless is also used to describe "rich-client" applications. Here, we limit our description to function-as-a-service. Please see this post by Mike Roberts for more information: https://martinfowler.com/articles/serverless.html#WhatIsServerless

CLOUD COST

The cloud is also changing how we think about costs for our infrastructure and deployments. No longer is the unit of measurement an entire datacenter or even a full rack of servers. Customers of cloud providers can focus on paying only for what they use while having the agility to scale when necessary.

In addition to positively impacting SDO performance, adopting cloud best practices improves organizations' visibility into the cost of running their technologies. Respondents who meet all essential cloud characteristics are 2.6 times more likely to be able to accurately estimate the cost to operate software. They are also twice as likely to be able to easily identify their most operationally expensive applications, and 1.65 times as likely to stay under their software operation budget.

BLACK BOX VS. TRANSPARENCY

Why can teams on the cloud better estimate and manage their costs? It is likely because the cloud provides better visibility into infrastructure usage and spend to developers and IT operations professionals. This increased visibility and awareness make it possible to change the way we architect and build our systems while also aligning incentives. While this variability can be initially confusing and overwhelming for those unused to this new financial model, teams can reap the benefits of efficient design by only paying for the compute resources they use.

In contrast, the data center in traditional environments is often a "black box," where information about processing and cycle cost is difficult or impossible to get. Additionally, the nature of capital expenses means that once infrastructure is purchased, there are no benefits for being aggressively efficient with design. In this regard, the capital expenses are a fixed cost—predictable and understood up-front, but rarely visible to the engineering team and impossible to avoid even if more efficient designs are deployed.

Some in finance may say that the cloud has not led to cost savings in the short-run, yet we know that it provides greater information transparency. How can this be? While the cloud provides transparent information about costs to the system owners, users do not pay for these costs unless there is a chargeback model or similar mechanism. This can lead to wildly variable costs that go unchecked, making cloud costs unpredictable. In these scenarios, teams that pay for infrastructure may prefer data centers because they are predictable, even though their visibility disincentivizes system users to build more efficient systems. We suggest organizations better align incentives so that system owners have both visibility to build more efficient systems, and the incentives to do so, by using chargeback or similar mechanisms.

While some workloads are better suited to on-prem environments (such as those that are steady or predictable), there are other benefits to using cloud infrastructure, such as the ability to leverage infrastructure-as-code.

TECHNICAL PRACTICES

Executing for maximum effect

Many organizations wanting to adopt DevOps look for a set of prescriptive steps or best practices to guide their journey. However, every organization is different and which practices to adopt depends on the current state of the organization—including the state of its technology, culture, and processes—and its short- and long-term goals.

The solution is to take a holistic approach, where you first work to understand the constraints in your current software delivery process with an eye to your short- and long-term outcomes in measurable terms. Then empower teams to decide how best to accomplish those outcomes— after all, they are the experts in their work and context.[14] Those who adopt this approach see more scalable and flexible solutions, and by not having to micromanage detailed execution plans, management can focus on high-level outcomes, allowing their organizations to grow. By focusing on designing and executing short-term outcomes that support the long-term strategy, teams are able to adjust to emergent and unanticipated

---

14 This approach derives from the work of Mike Rother based on his study of Toyota, which is described in detail at http://www-personal.umich.edu/~mrother/Homepage.html.

problems, outperforming their peers whose three- and five-year plans cannot be flexible and nimble enough to keep up with changes in customer demands, the technology landscape, or emergent security threats.[15]

While there is no "one size fits all" approach to improvement, we have observed some themes in our work helping organizations adopt DevOps. These themes are particularly relevant for companies looking to accelerate their transformation in the face of seemingly difficult and complex constraints.

Concurrent efforts at team and organization levels

Some capabilities are typically developed at the team level, while others—particularly in large organizations or organizations with strong hierarchical structures—often require organization-level efforts. These two streams—

team-level and organization-level—can and should proceed concurrently, as they often support each other.

CAPABILITIES RESEARCHED IN 2019[16]

Organization level	• Loosely coupled architecture • Clear change process • Code maintainability
Team level	• Continuous integration • Automated testing • Deployment automation • Monitoring • Trunk-based development
Both team and organizational level	• Use of cloud services • Disaster recovery testing

For example, creating a continuous integration platform that makes it easy for teams to get fast feedback on their automated tests can be a significant force-multiplier when used across several teams in an organization.

---

15 Providing teams with the capacity and resources on an ongoing basis is essential to the success of this approach.

16 For an exhaustive list of capabilities that drive improvements in SDO performance, we point the reader to _Appendix A_ in _Accelerate: The Science of Lean Software and DevOps_ (which summarizes the 2014 - 2017 State of DevOps Reports), the _2018 Accelerate State of DevOps Report_, and this Report.

Similarly, deployment automation at the team level will have little impact if the team's code can only be deployed together with that of other teams. This points to an architectural obstacle that must be resolved at the organizational level (which, in turn, is likely to require work from individual teams).

We will look at these capabilities in more detail, investigating them through the lens of team-level and organization-level capabilities.

Remember that our goal is improving our ability to deliver software, which we accomplish through technical practices in delivery and deployment we call **continuous delivery (CD)**. CD reduces the risk and cost of performing releases. Continuous delivery for the sake of continuous delivery is not enough if you want your organization to succeed, however. It must be done with an eye to organizational goals such as profitability, productivity, and customer satisfaction.

HOW WE MEASURED CONTINUOUS DELIVERY

Teams can deploy on-demand to production or to end users throughout the software delivery lifecycle.

Fast feedback on the quality and deployability of the system is available to everyone on the team and acting on this feedback is team members' highest priority.

Team-level technical capabilities

In previous years we found that **test automation** had a significant impact on CD. This year, we built upon prior years' research and found that automated testing positively impacts **continuous integration (CI)**. With automated testing, developers gain confidence that a failure in a test suite denotes an actual failure just as much as a test suite passing successfully means it can be successfully deployed. The ability to reproduce and fix failures, gather feedback from tests, improve test quality and iterate test runs quickly also ties into automated testing.

How is this different from previous research and what does it mean for you?

In previous years, we simply tested the importance of automated testing and CI, but didn't look at the relationship between the two. This year, we found that automated testing drives improvements in CI. It means that smart investments in building up automated test suites will help make CI better.

We revalidated that CI improves CD. For CI to be impactful, each code commit should result in a successful build of the software and a set of test suites being run. Automated builds and tests for a project should be run successfully every day.

We revalidated that deployment automation, trunk-based development[17], and monitoring impact CD. These capabilities may have dependencies on organization-level work, as we described for deployment automation. For example, teams can monitor their own code, but will not see full benefits if both application and infrastructure are not monitored and used to make decisions.

17 Our research shows that effective trunk-based development is characterized by fewer than three active branches and branches and forks having lifetimes of less than a day before being merged to master.

Organization-level technical capabilities

In contrast to capabilities that can be implemented and executed at the team level for quick impact, some capabilities benefit from organization-level coordination and sponsorship. Examples of these kinds of capabilities are those that involve decisions or design that span several teams, such as architecture or policy (e.g., change management).

This year's research revalidated the positive impact of **loosely coupled architecture** on CD. A loosely coupled architecture is when delivery teams can independently test, deploy, and change their systems on demand without depending on other teams for additional support, services, resources, or approvals, and with less back-and-forth communication. This allows teams to quickly deliver value, but it requires orchestration at a higher level.

OPEN SOURCE SOFTWARE DEVELOPMENT

Our research has focused on software development and delivery in an organizational context, where a team's full-time job is developing and delivering software, allowing members to coordinate their development and releases around a much tighter check-in and release cadence. We have found that trunk-based development with frequent check-in to trunk and deployment to production is predictive of performance outcomes.

But what about open source software development?

Open source projects have a different set of timelines and expectations since they are largely community-driven, with community members from around the world sending patches to projects when their schedule allows. Because open source projects must support collaboration with people around the world and across many organizations (including freelancers, hobbyists, and developers at all levels), open source project releases are cut in a different style than a continuous delivery software practice. They are typically cut from a branch at a specific point in time after significant testing.

Our research findings extend to open source development in some areas:

- Committing code sooner is better: In open source projects, many have observed that merging patches faster to prevent rebases helps developers move faster.

- Working in small batches is better: Large "patch bombs" are harder and slower to merge into a project than smaller, more readable patchsets since maintainers need more time to review the changes.

Whether you are working on a closed-source code base or an open source project, short-lived branches; small, readable patches; and automatic testing of changes make everyone more productive.

Architectural approaches that enable this strategy include the use of bounded contexts and APIs as a way to decouple large domains, resulting in smaller, more loosely coupled units. Service-oriented architectures are supposed to enable these outcomes, as should any true microservices architecture.[18] Architecture designs that permit testing and deploying services independently help teams achieve higher performance.

It takes a lot of code to run our systems: Facebook runs on 62 million lines of code (excluding backend code), the Android operating system runs on 12 to 15 million lines of code, and a typical iPhone app has 50,000 lines of code.[19] With the huge amount of code it takes to run our organizations, we wanted to investigate which code-related practices really help drive performance (or if they do at all), a construct we called **code maintainability**.

Our analysis found that code maintainability positively contributes to successful CD. Teams that manage code maintainability well have systems and tools that make it easy for developers to change code maintained by other teams, find examples in the codebase, reuse other people's code, as well as add, upgrade, and migrate to new versions of dependencies without breaking their code. Having these systems and tools in place not only contributes to CD, but also helps decrease technical debt, which in turn improves productivity, something you'll see in a later section.

Organizations that elevate code maintainability provide real advantages to their engineers. For example, managing dependencies is hard. Updating a dependency could open a rabbit hole to issues such as breaking API changes,

---

18 It's important to avoid premature decomposition of new systems into services, or overly fine-grained services, both of which can inhibit delivery performance.
    Martin Fowler covers this in *MonolithFirst* https://martinfowler.com/bliki/MonolithFirst.html.
19   https://www.businessinsider.com/how-many-lines-of-code-it-takes-to-run-different-software-2017-2

updating a transitive dependency, creating incompatible dependencies (for example, the diamond dependency issue), and breaking functionality. Tooling that can help avoid these errors or illuminate the consequences of code changes can improve design decisions and code quality for all engineers.

Debates about tools or code organization are easy to fall into. It's important to focus on outcomes: Are we enabling or preventing software performance and productivity?

USEFUL, EASY-TO-USE TOOLS FOR DEPLOYING SOFTWARE

Advanced users such as developers, testers, and sysadmins were previously neglected when considering the usability of their tooling. Sometimes management assumed that—as relative technology experts—the technologists could figure out any tool they were given. This isn't an uncommon mindset. In World War II, pilots were selected and trained based on their ability to operate overly complex cockpits. Then usability experts realized that complex work like piloting an aircraft was difficult enough. It was better to design a cockpit to be easy-to-use and understandable, and let pilots spend their attention safely piloting the aircraft.

Other times, usability needs are ignored because management assumes that technologists' needs are like those of regular end users. Today, we know that power users (such as engineers) often have special use cases, with unique design needs. Technologists also include broader skill sets and backgrounds—such as UX, infosec, and database engineers—as well as diverse abilities. Making tools that are accessible and easy-to-use is an important consideration for tool vendors.

With this in mind, in this year's research we studied the usability of the tools used to deploy software because technical practices that support software development and deployment are important to speed and stability.

The usefulness and ease-of-use of this deployment tooling is highly correlated with CI and CD. This makes sense, because the better our tools are suited to our work, the better we are able to do it. We also find this drives productivity, which you can read about on .

DISASTER RECOVERY TESTING

Every organization that runs mission-critical software systems should have a disaster recovery plan. But creating plans without testing them is like creating backups without also practicing restoring from backup regularly—that is to say, useless. We asked respondents which kinds of **disaster recovery testing** their organizations perform.

DISASTER RECOVERY TEST TYPES BY PERFORMANCE PROFILE

	Low	Medium	High	Elite	OVERALL
Table-top exercises that are not carried out on real systems	35%	26%	27%	30%	28%
Infrastructure (including datacenter) failover	27%	43%	34%	38%	38%
Application failover	25%	46%	41%	49%	43%
Simulations that disrupt production-like test systems (including failure injection such as degrading network links, turning off routers, etc.)	18%	22%	23%	29%	23%
Simulations that disrupt production systems (including failure injection such as degrading network links, turning off routers, etc.)	18%	11%	12%	13%	12%
Creating automation and systems that disrupt production systems on a regular, ongoing basis	9%	8%	7%	9%	8%

Not all tests are performed using production systems, which is a concern for two reasons. First, it's difficult (and often prohibitively expensive) to create comprehensive reproductions of production systems. Second, the types of incidents that bring down production systems are often caused by interactions between components that are operating within apparently normal parameters, which might not be encountered in test environments. As systems become more complex, these factors become more significant.

We asked how frequently organizations perform disaster recovery tests on production infrastructure:

- Simulations that disrupt production systems (including failure injection such as degrading network links, turning off routers, etc.)

- Infrastructure (including datacenter) failover

- Application failover

Only 40% of respondents perform **disaster recovery testing** at least annually using one or more of the methods listed. Organizations that conduct disaster recovery tests are more likely to have higher levels of service availability—that is, the ability for technology teams and organizations to make and keep promises and assertions about the software product or service they are operating.

Mike Garcia, Vice President of Stability & SRE at Capital One, says, "It's not enough to demonstrate you can deliver quickly on modern cloud technology. Especially in a heavily regulated industry like banking, we have obligations that require us to prove our level of resiliency in our responsibility to meet the needs of our customers. …In order to do that, we have had to advance beyond just demonstrating that we can failover en masse…The idea is to progressively show more advanced capabilities and automatic resiliency through more sophisticated chaos-testing techniques."

Organizations that work together cross-functionally and cross-organizationally to conduct disaster recovery exercises see improvements in more than just their systems. Because these tests pull together so many teams, they surface connections that many forget or don't realize exist, the exercises also improve and strengthen the processes and communication surrounding the systems being tested, making them more efficient and effective.

We also looked at whether organizations act on what they discover in disaster recovery testing exercises. Analysis shows that organizations that create and implement action items based on what they learn from disaster recovery exercises are 1.4 times more likely to be in the elite performing group.

LEARNING FROM DISASTER RECOVERY EXERCISES

Blameless post-mortems are an important aspect to support growth and learning from failure. This is well-documented in the literature and supported in our previous research, which showed that conducting blameless post-mortems contributes to a learning culture and an organizational culture that optimizes for software and organizational performance outcomes.

For a great read on disaster recovery exercises—from both the viewpoint of their benefits even in light of their expense, as well as a play-by-play from an SRE in the middle of an exercise—check out Weathering the Unexpected by Kripa Krishnan with Tom Limoncelli. [20]

In this ACM Queue article, Kripa Krishnan, a director at Google previously in charge of disaster recovery testing (DiRT) exercises, makes the following observation: "For DiRT-style events to be successful, an organization first needs to accept system and process failures as a means of learning. Things will go wrong. When they do, the focus needs to be on fixing the error instead of reprimanding an individual or team for a failure of complex systems."

20   https://queue.acm.org/detail.cfm?id=2371516

CHANGE MANAGEMENT

Making changes to software and production systems is often complex and bureaucratic. Two factors are responsible for much of this complexity: the need for coordination between teams, and requirements of regulatory control, particularly in financial services, healthcare, and government. While the complexities involved in implementing regulatory control requirements are beyond the influence of leadership and practitioners, we can influence the role that team coordination plays in change management—and that role is changing.

For example, segregation of duties, which states that changes must be approved by someone other than the author, is often required by regulatory frameworks.[21] While we agree that no individual should have end-to-end control over a process (the intent of this control), there are lightweight, secure ways to achieve this objective that don't suffer the same coordination costs as heavyweight approaches.

---

21 Examples include the Payment Card Industry Data Security Standard (PCI-DSS) and the NIST Risk
    Management Framework used by US Federal Government agencies.

One approach is to require every change be approved by someone else on the team as part of code review, either prior to commit to version control (as part of pair programming) or prior to merge into master.

This can be combined with automated thresholds that bound changes. For example, you may implement checks to not allow developers to push a change (even with peer review) that will increase compute or storage costs over a certain threshold. This lightweight, straightforward-to-implement process presents a clear opportunity for practitioners to improve change management. Some proponents, supported by IT service management (ITSM) frameworks, claim that formal change approval processes lead to more stability, and point out that these have been the norm in industry for decades.

IMPLEMENTING SEGREGATION OF DUTIES

Using code review to implement segregation of duties requires that all changes to production systems should be recorded in a change management system that lists the change along with the person or people who authored it, and logs authenticated approval events. If you're using version control as the source of truth for all changes to your systems (a technique from the paradigm known as "infrastructure as code" or "gitops"), you simply need to be able to record who approved each change. This has the added benefit of making the review and submit process highly auditable. Our previous research also shows that comprehensive use of version control, including for system configuration and scripting, drives performance.

Others, backed by lean and agile philosophies, argue that more streamlined change approvals lead to faster feedback, better information flow, and better outcomes. To investigate these hypotheses, we created two new constructs— one that captures a lightweight, clearly understood change approval process, and another that captures a more formal, **heavyweight change approval process**—and tested their impact on software delivery performance.

Heavyweight change process

We found that formal change management processes that require the approval of an external body such as a change advisory board (CAB) or a senior manager for significant changes have a negative impact on software delivery performance. Survey respondents were 2.6 times more likely to be low performers if their organization had this kind of formal approval process in place. This expands on our previous research, which found that heavyweight change approvals process were negatively correlated with change failure rates.[22]

The motivation behind the heavyweight change management processes proposed by ITSM frameworks is reducing the risk of releases. To examine this, we investigated whether a more formal approval process was associated with lower change fail rates and we found no evidence to support this hypothesis, consistent with earlier research.[23] We also examined whether introducing more approvals results in a slower process and the release of larger batches less frequently, with an accompanying higher impact on the production system that is likely to be associated with higher levels of risk and thus

22,23 Velasquez, N., Kim, G., Kersten, N., & Humble, J. (2014). State of DevOps Report: 2014. *Puppet Labs*.

higher change fail rates. Our hypothesis was supported in the data. This has important implications for organizations working to reduce risk in their release process: Organizations often respond to problems with software releases by introducing additional process and more heavyweight approvals. Analysis suggests this approach will make things worse.

We recommend that organizations move away from external change approval because of the negative effects on performance. Instead, organizations should "shift left" to peer review-based approval during the development process. In addition to peer review, automation can be leveraged to detect, prevent, and correct bad changes much earlier in the delivery lifecycle. Techniques such as continuous testing, continuous integration, and comprehensive monitoring and observability provide early and automated detection, visibility, and fast feedback. In this way, errors can be corrected sooner than would be possible if waiting for a formal review.

WHAT HAPPENS TO THE CAB IN THE CONTINUOUS DELIVERY PARADIGM?

Continuous delivery offers a superior risk management approach compared to traditional change management processes, but there is still an important role for the CAB. As organizations become more complex, facilitating notification and coordination among teams is increasingly critical. Our previous research has established that fostering information flow is essential to developing a high-performance, mission-driven culture. Since approving each individual change is impossible in practice in the continuous paradigm, the CAB should focus instead on helping teams with process-improvement work to increase the performance of software delivery. This can take the form of helping teams implement the capabilities that drive performance by providing guidance and resources. CABs can also weigh in on important business decisions that require a trade-off and sign-off at higher levels of the business, such as the decision between time-to-market and business risk.

You'll note the new role of the CAB is strategic. By shifting detailed code review to practitioners and automated methods, time and attention of those in leadership and management positions is freed up to focus on more strategic work. This transition, from gatekeeper to process architect and information beacon, is where we see internal change management bodies headed, and is consistent with the practices of organizations that excel at software delivery performance.

Clear change process

While moving away from traditional, formal change management processes is the ultimate goal, simply doing a better job of communicating the existing process and helping teams navigate it efficiently has a positive impact on software delivery performance. When team members have a clear understanding of the process to get changes approved for implementation, this drives high performance. This means they are confident they can get changes through the approval process in a timely manner and know the steps it takes to go from "submitted" to "accepted" every time for all the types of changes they typically make. Survey respondents with a **clear change process** were 1.8 times more likely to be in elite performers

In our work with large organizations, change management is consistently one of the biggest constraints. Removing it requires work at multiple levels.

Teams can implement continuous integration, continuous testing, and peer review to find bad changes as quickly as possible while also satisfying segregation of duties. And only our technical practitioners have the power to build and automate the change management solutions we design, making them fast, reliable, repeatable, and auditable.

Leaders at every level should move away from a formal approval process where external boards act as gatekeepers approving changes, and instead move to a governance and capability development role. After all, only managers have the power to influence and change certain levels of organizational policy. We have seen exponential improvements in performance—throughput, stability, and availability—in just months as a result of technical practitioners and organizational leaders working together.

CULTURE OF PSYCHOLOGICAL SAFETY

Culture is often lauded as the key to DevOps and technology transformations by practitioners and champions who lead efforts in organizations. Indeed, Davis and Daniels cite culture as a key factor in successful and scalable technology efforts in their book Effective DevOps.[24] Our own research has found that an organizational culture that optimizes for information flow, trust, innovation, and risk-sharing is predictive of SDO performance.[25]

Research from a large two-year study at Google found similar results:[26] that high-performing teams need a culture of trust and psychological safety, meaningful work, and clarity. This team environment allows members to take calculated and moderate risks, speak up, and be more creative. Some have wondered if these results can also be true outside of Google. Is this kind of culture beneficial to the wide mix of enterprises, tools, and engineering skills we see in other organizations? Or does it only hold true for the engineers that pass Google's notoriously rigorous interviews, in a large enterprise, supported by huge infrastructure and only a certain type of code base?

---

24 Davis, J., & Daniels, R. (2016). Effective DevOps: building a culture of collaboration, affinity, and tooling at scale. O'Reilly Media, Inc.
25 This research was based on an organizational culture framework originally proposed by the sociologist Dr. Ron Westrum.
    This model is outlined in the 2018 Accelerate State of DevOps Report.
26 https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/

To answer that question, we shifted our measure of culture to the questions that the Project Aristotle team included in its research.[27] Our analysis found that this **culture of psychological safety** is predictive of software delivery performance, organizational performance, and productivity. Although Project Aristotle measured different outcomes, these results indicate that teams with a culture of trust and psychological safety, with meaningful work and clarity, see significant benefits outside of Google.

1	**PSYCHOLOGICAL SAFETY** Team members feel safe to take risks and be vulnerable in front of each other
2	**DEPENDABILITY** Team members get things done on time and meet Google's high bar for excellence
3	**STRUCTURE AND CLARITY** Team members have clear roles, plans, and goals
4	**MEANING** Work is personally important to team members
5	**IMPACT** Team members think their work matters and creates change

re:Work

TECHNOLOGY'S IMPACT ON ORGANIZATIONAL PERFORMANCE

The ability to deliver software rapidly and reliably and provide high levels of availability is a powerful tool. It enables organizations to easily and quickly prototype new products and features and test the impact on users without impacting existing users. It also allows organizations to keep up with compliance and regulatory changes, and deliver critical software patches and updates necessary for security quickly and reliably. If leveraged effectively, organizations that can achieve high levels of SDO performance should be able to better respond to technological change and shifts in the market and create superior products and services. This, in turn, helps organizations better achieve their desired organizational outcomes, both commercial and non-commercial. **Our analysis this year shows elite performers are twice as likely to meet or exceed their organizational performance goals.**

The organizational performance measures we use are derived from academic literature and capture both commercial[28] and non-commercial[29] goals, including:

- Profitability
- Productivity
- Market share
- Number of customers
- Quality of products or services

- Operating efficiency
- Customer satisfaction
- Quality of products or services provided
- Achieving organizational or mission goals

As with last year, our second-order construct of SDO performance predicts organizational performance, and does so better than software delivery performance or availability do alone.

28 Widener, S. K. (2007). An empirical analysis of the levers of control framework. *Accounting, Organizations and Society*, 32(7-8), 757-788.
29 Cavalluzzo, K. S., & Ittner, C. D. (2004). Implementing performance measurement innovations: evidence from government. *Accounting, Organizations and Society,* 29(3-4), 243-267.

27 Many thanks to the Project Aristotle team for sharing their research instrument.

HOW DO WE IMPROVE PRODUCTIVITY?

Another important goal in teams and organizations is improving productivity to get more value out of your transformation and your employees. This marks the first year we investigate productivity: how organizations can support it with smart investments in tools and information, how technical debt interrupts it, and how it affects employee work/life balance and burnout.

IMPROVING PRODUCTIVITY

Most agree that productivity is important: Productive engineers are able to do their work more efficiently, giving them more time to re-invest into other work,[30] such as documentation, refactoring, or doing more of their core function to deliver additional features or build out additional infrastructure.[31]

But what is productivity, and how should we measure it? Productivity cannot be captured with a simple metric such as lines of code, story points, or bugs closed; doing so results in unintended consequences that sacrifice the overall goals of the team.[32] For example, teams may refuse to help others because it would negatively impact their velocity, even if their help is important to achieve organizational goals.

---

30 We point the reader to a discussion of productivity by noted economist Hal Varian https://www.wsj.com/articles/silicon-valley-doesnt-believe-u-s-productivity-is-down-1437100700

31 We discuss the benefits of reinvesting the time savings from productivity gains (vs. seeing them as cost savings to be eliminated) in our 2017 ROI white paper "Forecasting the Value of DevOps Transformations." This is not a new idea; economists have investigated this and Dr Varian addresses it in his WSJ article cited above. cloud.google.com/devops
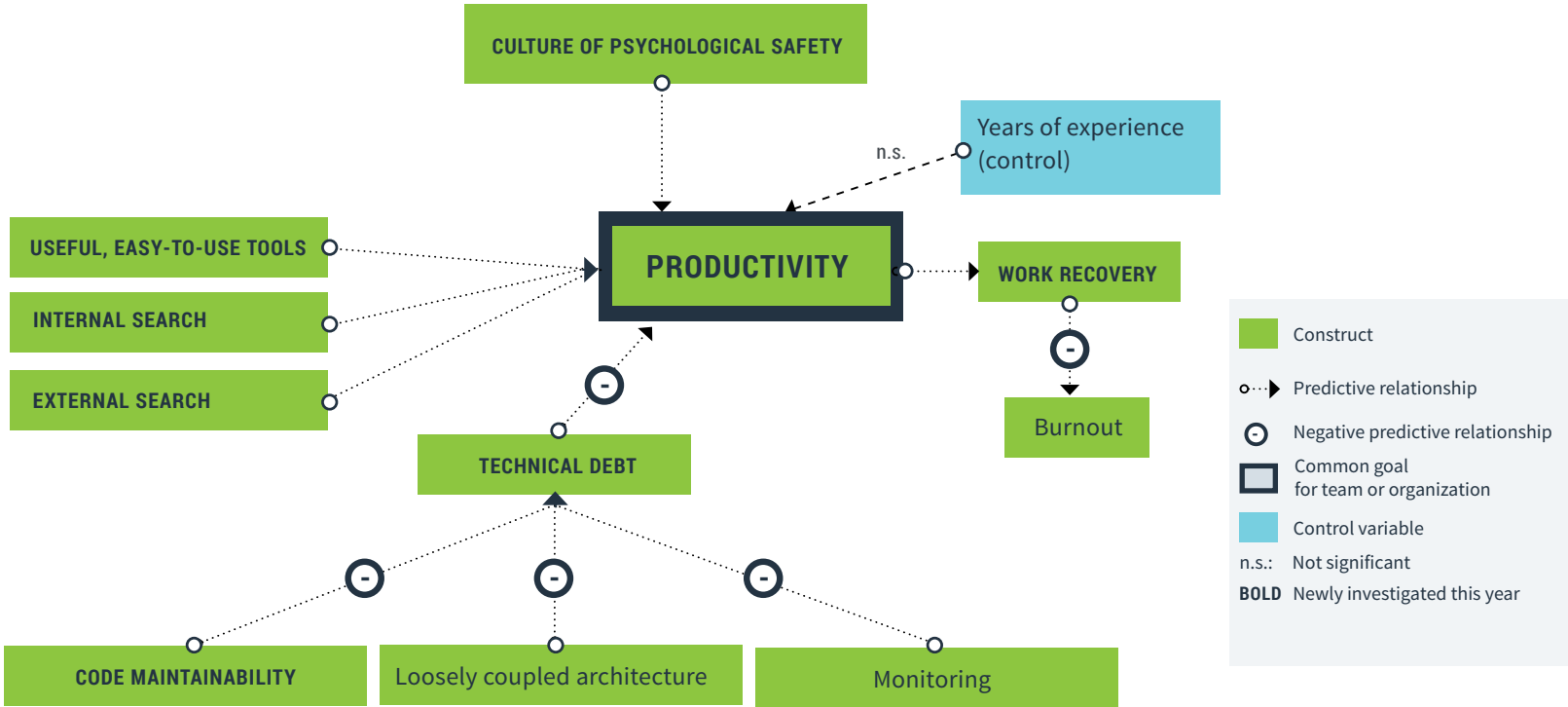
32 This can also be summarized by Goodhart's Law, which states that "When a measure becomes a target, it ceases to be a good measure." By focusing only on easy-to-measure, local metrics, teams often set those as their targets, at the expense of global goals that truly further organizational outcomes. See Chapter 2 of *Accelerate: The Science of Lean Software and DevOps* for more detail.

Researchers have discussed this topic at length, and most have come to the same conclusion:

Productivity is the ability to get complex, time-consuming tasks completed with minimal distractions and interruptions.

Many of us describe this as getting into a good work flow or rhythm.

To use this model, locate the goal you want to improve in the figure, and then identify the capabilities that impact it. For example, if your goal is to reduce technical debt, these capabilities are code maintainability, having a loosely coupled architecture, and monitoring.

USEFUL, EASY-TO-USE TOOLS

Useful and easy-to-use tools are now considered a must-have for consumer technologies, but these obvious characteristics are often overlooked among technology professionals who assume they are experts and can make any tool or technology work. (Or because those purchasing tools for these groups assume usability is less important for technologists, or are optimizing for other factors such as cost, licensing terms, or vendor management.) In fact, the opposite is true: When building complex systems and managing business-critical infrastructure, tools are even more important because the work is more difficult.

We focused on tools used in deploying software through the CI/CD and test automation toolchain because they are at the heart of DevOps. We found that these two attributes drive productivity:

- How easy it is to use the toolchain (including straightforward and easy interactions and operation)

- How useful the toolchain is in accomplishing job-related goals

HIGHEST PERFORMING ENGINEERS

1.5 TIMES MORE
likely to have easy-to-use tools

TOOL USAGE BY PERFORMANCE PROFILE

	Low	Medium	High	Elite
A mix of proprietary tools, open source, and commercial off-the-shelf (COTS) software	30%	34%	32%	33%
Mainly open source and COTS, heavily customized	17%	8%	7%	10%
Mainly open source and COTS, with little customization	14%	21%	18%	20%
Primarily COTS packaged software	8%	12%	8%	4%
Primarily developed in-house and proprietary to my organization	20%	6%	5%	6%
Primarily open source, heavily customized	6%	7%	5%	12%
Primarily open source, with little customization	5%	12%	24%	15%

And what do those tools look like? We dug into the data by performance profile and saw some interesting patterns:

- The strongest concentration of fully proprietary software is seen in low performers, while the lowest concentration is seen among high and elite performers. Proprietary software may be valuable, but it comes at great cost to maintain and support. It's no surprise that the highest performers have moved away from this model.

- There is a relatively equal concentration of commercial off-the-shelf (COTS) software with little customization. Some may wonder why high performers can use COTS and still be high performers, especially if the message continues to be that "software is eating the world" and we must be creating our own software.

59

Accelerate: State of DevOps 2019 | How Do We Improve Productivity?

AUTOMATION AND INTEGRATION BY PERFORMANCE PROFILE

	Low	Medium	High	Elite
Automated build	64%	81%	91%	92%
Automated unit tests	57%	66%	84%	87%
Automated acceptance tests	28%	38%	48%	58%
Automated performance tests	18%	23%	18%	28%
Automated security tests	15%	28%	25%	31%
Automated provisioning and deployment to testing environments	39%	54%	68%	72%
Automated deployment to production	17%	38%	60%	69%
Integration with chatbots / Slack	29%	33%	24%	69%
Integration with production monitoring and observability tools	13%	23%	41%	57%
None of the above	9%	14%	5%	4%

33 Martin Fowler, MartinFowler.com, UtilityVsStrategicDichotomy.
https://martinfowler.com/bliki/UtilityVsStrategicDichotomy.html
34 This is a site reliability engineering (SRE) best practice: reduce toil, which is work without productivity.

As Martin Fowler outlines,[33] companies should be thoughtful about which software is strategic and which is merely utility. By addressing their utility needs with COTS solutions and minimizing customization, high performers save their resources for strategic software development efforts.

We also see that elite performers automate and integrate tools more frequently into their toolchains on almost all dimensions. Although automation may be seen as too expensive to implement (we often hear, "I don't have time or budget to automate—it's not a feature!"), automation is truly a sound investment.[34] It allows engineers to spend less time on manual work, thereby freeing up time to spend on other important activities such as new development, refactoring, design work, and documentation. It also gives engineers more confidence in the toolchain, reducing stress in pushing changes.

Technical professionals and tools

Our work in 2017 found that empowered teams who make their own decisions about tools and implementations contribute to better software delivery performance. In this year's research, we see that given the opportunity, high performers choose useful and usable tools, and these kinds of tools improve productivity.

This has important implications for product design. Products that have both utility and usability are more likely to be adopted by technology professionals, and when they are used, have better outcomes. These kinds of tools should be prioritized by industry leaders. It's not enough to deliver products that are feature complete; they also need to be usable to be adopted and deliver value during a DevOps transformation.

PRODUCTIVITY, BURNOUT, AND JUGGLING WORK

We wondered if the amount of juggling work would be significantly different among our highest and lowest performers—after all, productivity is the ability to get work done and feel like you are in "a flow."

To capture this, we asked respondents a few questions:

- How many roles they juggle or different types of work do they do regardless of their official job title

- How many projects they switched between in a day

- How many projects they were working on overall

Surprisingly, we did not detect significant differences between low, medium, high, and elite performers. Therefore, we cannot conclude that how well teams develop and deliver software affects the number of roles and projects that respondents juggle. There is no such thing as "push through this phase and it will get significantly better." Instead, we should take steps to make our work sustainable. That is done through process improvement work and automation, which will reduce toil and make the work repeatable, consistent, fast, scalable, and auditable. It will also free us up to do new, creative work.

INTERNAL AND EXTERNAL SEARCH

Finding the right information to help solve a problem, debug an error, or find a similar solution quickly and easily can be a key factor in getting work done and maintaining the flow of work. This is especially true in today's tech environment, which is comprised of increasingly complex systems. We found that having access to information sources supports productivity. These information sources come in two categories: internal and external search.

- **Internal search:** Investments that support document and code creation as well as effective search for company knowledge bases, code repositories, ticketing systems, and other docs contribute to engineering productivity. Those who used internal knowledge sources were 1.73 times more likely to be productive. Providing developers, sysadmins, and support staff with the ability to search internal resources allows them to find answers that are uniquely suited to the work context (for example, using "find similar" functions) and apply solutions faster. In addition, internal knowledge bases that are adequately supported and fostered create opportunities for additional information sharing and knowledge capture.

**USE OF
INTERNAL SEARCH**

**1.73
TIMES MORE**
likely to be productive

**USE OF
EXTERNAL SEARCH**

**1.67
TIMES MORE**
likely to be productive

For example, people may pose questions when they find solutions that almost fit, prompting answers and discussion from the internal community, which feeds additional information into the knowledge base. If the organization has invested in systems that can easily search across all types of information and data, the culture can contribute to a "virtuous cycle" of knowledge sharing. Some organizations are leveraging machine learning technologies to identify and suggest candidate solutions to internal search as well.

- **External search:** These include external sources such as search engines and Stack Overflow. Our analysis found that those who used external search in their work were 1.67 times more likely to report feeling productive in their work. External search is important because these technologies provide strong communities for learning and growing (and recruiting, an important side benefit) and provide support for the use and adoption of public cloud and open source tooling. That is, leveraging commonly used external tools and systems with a strong user community and good ecosystem allows tech professionals to troubleshoot with the world, while proprietary and home-grown implementations only allow experts within an organization to weigh in on possible solutions. We see support for this in the data. In our 2018 research, elite performers leveraged and planned to expand their use of open source. In this year's research, we see that eite performers use more open source tooling and low performers have the highest use of proprietary data (see page 59); these technology choices are bound to have an impact on productivity.

TECHNICAL DEBT

Technical debt was introduced in 1992 by Ward Cunningham[35] to describe what happens when we fail to adequately maintain what he calls "immature" code.

> *Although immature code may work fine and be completely acceptable to the customer, excess quantities will make a program unmasterable, leading to extreme specialization of programmers and finally an inflexible product. Shipping first-time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a standstill under the debt load of an unconsolidated implementation.*

In today's complex systems, technical debt can occur in scripts, configuration files, and infrastructure as well as application code. Technical debt includes code or systems with:

- Known bugs that go unfixed in favor of new features

- Insufficient test coverage

- Problems related to low code quality or poor design

- Code or artifacts that aren't cleaned up when no longer used

- Implementations that the current team doesn't have expertise in, and therefore can't effectively debug or maintain

- Incomplete migration

- Obsolete technology

- Incomplete or outdated documentation or missing comments

---

35 http://c2.com/doc/oopsla92.html

We found that technical debt negatively impacts productivity, that respondents with high technical debt were 1.6 times less productive, and that the highest performers were 1.4 times more likely to have low technical debt.

Coping with technical debt

Ward Cunningham states that "the best antidote [to changing systems] is more complete familiarity with the product and its implementation." When engineers can understand changes, they can accept them. Today's complex infrastructures and distributed systems make it impossible for engineers to maintain a mental model of the complete state of the system. In addition, supporting these complex systems has led to more specialized professionals who cannot have complete familiarity with the entire system.[36]

We can help engineers build mental models by architecting for flexible, extensible, and visible systems to reduce technical debt.

---

36 In order to have complete visibility into the system, one would have to be a full stack developer. Aside from the obvious definitional challenge—What is full stack? Chips to CSS?—many in the industry agree that a full stack developer isn't possible or desirable.

ACTIVELY REDUCING TECHNICAL DEBT

How can we actually reduce technical debt and not just cope with it? One approach is refactoring. Refactoring is a "disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior," and Martin Fowler points out that refactoring should be part of daily work. Better tooling with robust refactoring support built in are also important. In fact, many large organizations have invested in tools for doing code refactors across their code base; for example, Facebook open sourced its tool fastmod and Google has open sourced ClangMR.

CULTURE OF PSYCHOLOGICAL SAFETY

A **culture that values psychological safety**, trust, and respect contributes to productivity by letting employees focus on solving problems and getting their work done rather than politics and fighting. This echoes work by other researchers; as we discuss in the section earlier, a study by Google found that this same kind of culture leads to more effective teams.

PRODUCTIVITY AND OPEN SOURCE PROJECTS

The findings on productivity can also apply to open source projects. Contributors can go from zero to productive on open source projects faster if the documentation for how to contribute is up-to-date, simple, and consistent with other open source projects. A large project with a custom and outdated contribution process will have a hard time getting new contributors since the path to entry is too complex. Pair a complicated contribution process with technical debt and your would-be contributors will view your project as "read-only." Applying all the same best practices mentioned here to your open source project will get new contributors into the flow much faster and grow your community in the most productive way.

ADDITIONAL BENEFITS OF IMPROVED PRODUCTIVITY

The benefits to the team and organization from higher productivity
are usually obvious: more work gets done, so we deliver more value.
But what about benefits to the people doing the work?

Work Recovery

Research shows that productivity has a positive impact on **work recovery**,
and we find support for that in our data as well. Work recovery is the ability
to cope with work stress and detach from work when we are not working.
Some call this "leaving work at work," and research shows that people who
can detach from work have better well-being and handle work-related
stress better. The reverse of this is also important: Feeling overworked leads
to difficulty detaching, which leads to burnout and lower life satisfaction.[37]

Burnout

Burnout has been recognized by the World Health Organization as a
condition that results from unmanaged chronic workplace stress,[38]
and it is more than just being tired. Burnout is a combination of exhaustion,

---

37 Sonnentag, S., & Fritz, C. (2015). Recovery from job stress: The stressor-detachment model as an integrative framework. *Journal of Organizational Behavior*, 36(S1), S72-S103.
    https://onlinelibrary.wiley.com/doi/abs/10.1002/job.1924
38 World Health Organization, Burn-out an "occupational phenomenon": *International Classification of Diseases*.
    https://www.who.int/mental_health/evidence/burn-out/en/

cynicism, and inefficacy at work. It is often seen in complex, high-risk work in hospitals, air traffic control, and technology,[39] and research shows that stressful jobs can be as bad for physical health as secondhand smoke[40] and obesity.[41] In extreme cases, burnout can lead to family issues, clinical depression, and even suicide.

In this year's research, we found that work recovery can reduce burnout, which confirms other research. We also revalidated previous years' research and found that good technical practices and improved process (in the form of clear change management) can reduce burnout. We see that the highest performers are half as likely to report feeling burned out. Said another way, low performers are twice as likely to report feeling burned out.

---

39 While some claim technology is not high-risk work, the stresses of pushing the wrong code can be very real: Errors that have the potential to be very public or have life-changing implications (such as writing software that supports hospitals or healthcare systems) is the reality for many in technology.

40 Goh, J., Pfeffer, J., Zenios, S. A., & Rajpal, S. (2015). Workplace stressors & health outcomes: Health policy for the workplace. *Behavioral Science & Policy*, 1(1), 43-52.

41 Chandola, T., Brunner, E., & Marmot, M. (2006). Chronic stress at work and the metabolic syndrome: prospective study. *BMJ,* 332(7540), 521-525.

HOW TO SUPPORT WORK RECOVERY

Productivity has a positive impact on work recovery. Since this can be a key to reducing stress and burnout, how can we support it?

Research shows that there are five keys to promote work recovery:

1 **Psychological detachment** is the ability to stop thinking of work outside of work. To promote this, consider electronic barriers that help keep work stress at work.

2 **Relaxation** is not a luxury that we indulge on vacation; it's a key component of productivity. Make a routine out of giving yourself time to recover.

3 **Mastery** of a skill outside work promotes a positive outlook, reduces stress, and promotes healthy relationships. Practice skills outside of work that bring you joy.

4 A lack of **control** causes stress. Balance a lack of control over work demands with active pursuits outside of work, which you can control.

5 Foster a culture that encourages stepping away from work. Management in particular can set the tone that people are expected to go home and not work on evenings and weekends. Practitioners can support this culture by disconnecting when they're not at work, and encouraging colleagues to do the same.*

* A culture of psychological safety is strongly correlated with work recovery, and can help support it. This is true because team members can feel safe to talk about stress, workloads, and taking time off when they head home. We didn't test a predictive relationship because our measure of culture didn't specifically ask about "supporting and encouraging taking time away from work," and those kinds of cultures have been shown to increase work recovery.

HOW DO WE TRANSFORM: WHAT REALLY WORKS

We are often asked how companies spread new ways of work throughout their organization. Constant restructuring and reorganizing isn't sustainable due to its short-term negative impact on productivity, and we know of several organizations who have executed major transformations that haven't undergone a reorg. While there isn't a golden path to success, one thing is clear: A DevOps transformation is not a passive phenomenon. This year we sought to identify the most common approaches for spreading DevOps best practices throughout an organization.

TRANSFORM: WHAT REALLY WORKS

We asked respondents to share how their teams and organizations spread DevOps and Agile methods.

Respondents could select one or more of the following approaches:*

- Training Center (sometimes referred to as a DOJO)
- Center of Excellence
- Proof of Concept but Stall
- Proof of Concept as a Template
- Proof of Concept as a Seed
- Communities of Practice
- Big Bang
- Bottom-up or Grassroots
- Mashup

These items were created based on commonly used approaches that we have observed across the industry. Once generated, we asked a representative team of subject matter experts to review the list. They gave us feedback and helped us refine the options for clarity and comprehensiveness.

* Refer to Appendix B for detailed descriptions of each approach

We examined the data across our SDO performance clusters to examine the effectiveness of each strategy. It is not a perfect proxy, but it does provide a glimpse into what the highest and lowest performers are doing to scale their technology transformations.

Mashups are commonly reported in this sample at 40%, but they lack sufficient funding and resources in any particular investment. We caution that without a strategy to guide a technology transformation, organizations will often make the mistake of hedging their bets and suffer from "death by initiative": identifying initiatives in too many areas, which ultimately leads to under-resourcing important work and dooming them all to failure. Instead, it is best to select a few initiatives and dedicate resources to ensure their success (time, money, and executive and champion practitioner sponsorship).[42]

High performers favor strategies that create community structures at both low and high levels in the organization, likely making them more sustainable and resilient to reorgs and product changes. The top two strategies employed are Communities of Practice and Grassroots, followed by Proof of Concept (PoC) as a Template (a pattern where the PoC copies) and PoC as a Seed.

Low performers tend to favor Training Centers (also known as DOJOs) and Centers of Excellence (CoE)—strategies that create more silos and isolated expertise. They also attempt PoCs, but these generally stall and don't see success.

Some strategies see common patterns among all performance profiles: All profiles report adopting and supporting a mix of strategies.

---

42 This is much like the constraints model of continuous improvement we outline earlier: Set short- and long-term goals, identify the key areas that need the most improvement to achieve those goals, and allocate resources accordingly.

No profiles report strong use of a Big Bang strategy—though low performers use this the most often (19% of the time)—and that's probably for the best. In our experience, this is an incredibly difficult model to execute and should only be attempted in the most dire of situations, when a "full reset" is needed. In the Big Bang, everyone needs to be on board for the long-haul, with resources dedicated for a multi-year journey. This may explain why this method is seen most often among our low performers.

Why aren't CoEs and Training Centers recommended?

In general, Centers of Excellence (CoEs) are not recommended because they centralize expertise in one group. This creates several problems. First, the CoE is now a bottleneck for the relevant expertise for the organization and this cannot scale as demand for expertise in the organization grows. Second, it establishes an exclusive group of "experts" in the organization, in contrast to an inclusive group of peers who can continue to learn and grow together. This exclusivity fosters bad norms and behaviors and can chip away at healthy organizational cultures. Finally, the experts are removed from doing the work. They are able to make recommendations or establish generic "best practices" but the path from the generic learning to the implementation of real work is left up to the learners. For example, experts will build a workshop on how to containerize an application, but they rarely or never actually containerize applications. This disconnect between theory and hands-on practice will eventually threaten their expertise.

While some see success in Training Centers, they require dedicated resources and programs to execute both the original program and sustained learning. Many companies have set aside incredible resources to make their Training Programs effective: They have entire buildings dedicated to a separate, creative environment, and staff devoted to create training materials and assess progress. Additional resources are then needed to assure that the learning is sustained and propagated throughout the organization. The organization has to provide support for the teams that attended the Training Center, to help ensure their skills and habits are continued back in their regular work environments, and that old work patterns aren't resumed. If these resources aren't in place, organizations risk all of their investments going to waste. Instead of a Center where teams go to learn new technologies and

processes to spread to the rest of the organization, new habits stay in the Center, creating another silo, albeit a temporary one. There are also similar limitations as in the CoE: If only the Training Center staff (or other, detached "experts") are creating workshops and training materials, what happens if they never actually do the work?

HEATMAP OF DEVOPS TRANSFORMATION STRATEGIES BY PERFORMANCE PROFILE

	Low	Medium	High	Elite
Training Center	27%	21%	18%	14%
Center of Excellence	34%	34%	20%	24%
Proof of Concept but Stall	41%	32%	20%	16%
Proof of Concept as a Template	16%	29%	29%	30%
Proof of Concept as a Seed	21%	24%	29%	30%
Communities of Practice	24%	51%	47%	57%
Big Bang	19%	19%	11%	9%
Bottom-up or Grassroots	29%	39%	46%	46%
Mashup	46%	42%	34%	38%

Scaling strategies that work

We conducted an additional cluster analysis to understand the strategies used most often by high and elite performers, and identify four model patterns that emerge.

- **Community Builders:** This group focuses on Communities of Practice, Grassroots, and PoCs (as a Template and as a Seed, as described earlier). This occurs 46% of the time.

- **University:** This group focuses on education and training, with the majority of their efforts going into Centers of Excellence, Communities of Practice, and Training Centers. We see this pattern only 9% of the time, suggesting that while this strategy can be successful, it is not common and requires significant investment and planning to ensure that learnings are scaled throughout the organization.

- **Emergent:** This group has focused on Grassroots efforts and Communities of Practice. This appears to be the most hands-off group and appears in 23% of cases.

- **Experimenters:** Experimenters appeared in 22% of cases. This group has high levels of activity in all strategies except Big Bang and DOJOs—that is, all activities that focus on community and creation. They also include high levels in PoC but Stall, and because they are able to leverage this activity and remain high performers suggests they use this strategy to experiment and test out ideas quickly.

With these four patterns in mind, we can begin to strategize how to organize a DevOps transformation. High and elite performers have started to develop strategies for scaling that organizations can choose from to mirror those efforts and improve their own performance.

FINAL THOUGHTS

Every decade has its own trendy software methodology. While they all seem to feel better, history proves them to be ineffective. However, we see continued evidence that DevOps delivers value, and for six consecutive years, we have statistically verified key capabilities and practices that help organizations improve their software development and delivery using DevOps methods.

DevOps is not a trend, and will eventually be the standard way of software development and operations, offering everyone a better quality of life.

We thank everyone who contributed to this year's survey, and hope our research helps you and your organization build better teams and better software—while also leaving work at work.

METHODOLOGY

Our rigorous methodology goes beyond reporting raw numbers and looks at the predictive relationships between SDO performance, organizational performance, technical practices, cultural norms, and productivity. In this section, we describe our analysis methods, as well as how we enlisted survey respondents and how we designed our questions, models, and constructs. For more detail, we point you to Part II of our book *Accelerate: The Science of Lean Software and DevOps.*

We welcome questions about our survey methodology at dora-data@google.com.

Research design

This study employs a cross-sectional, theory-based design. This theory-based design is known as inferential, or inferential predictive, and is one of the most common types conducted in business and technology research today. Inferential design is used when purely experimental design is not possible and field experiments are preferred—for example, in business, when data collection happens in complex organizations, not in sterile lab environments—and companies won't sacrifice profits to fit into control groups defined by the research team.

Target population and sampling method

Our target population for this survey was practitioners and leaders working in, or closely with, technology work and transformations and especially those familiar with DevOps. Because we don't have a master list of these people—we can describe them, but we don't know exactly where they are, how to find them, or how many of them exist—we used snowball sampling to obtain respondents. This means we promoted the survey via email lists, online promotions, and social media, and also asked people to share the survey with their networks, growing the sample like a snowball rolling down a hill collects additional snow. Our sample is likely limited to organizations and teams that are familiar with DevOps, and as such, may be doing some of it. A key to overcoming limitations in snowball sampling is to have a diverse initial sample. We accomplished this by leveraging our own contact lists as well as those of our sponsors for our initial sample, resulting in demographics and firmographics that largely match industry trends.

Creating latent constructs

We formulated our hypotheses and constructs using previously validated constructs wherever possible. When we needed to create new constructs, we wrote them based on theory, definitions, and expert input. We then took additional steps to clarify intent and wording to ensure that data collected from the final survey would have a high likelihood of being reliable and valid.[43] We used Likert-type[44] questions for construct measurement, which make it possible to perform more advanced analyses.

---

43 We used Churchill's methodology: Churchill Jr, G. A. "A paradigm for developing better measures of marketing constructs," *Journal of Marketing Research* 16:1, (1979), 64–73.

44 McLeod, S. A. (2008). Likert scale. Retrieved from www.simplypsychology.org/likert-scale.html

Statistical analysis methods

- **Cluster analysis.** We use cluster analysis to identify our software delivery performance profiles and scaling approaches used by high performers. In this approach, those in one group are statistically similar to each other and dissimilar from those in other groups, based on our performance behaviors of throughput and stability: deployment frequency, lead time, time to restore service, and change fail rate. A solution using Ward's method[45] was selected based on (a) change in fusion coefficients, (b) number of individuals in each cluster (solutions including clusters with few individuals were excluded), and (c) univariate F-statistics.[46] We used a hierarchical cluster-analysis method because it has strong explanatory power (letting us understand parent-child relationships in the clusters) and because we did not have any industry or theoretical reasons to have a predetermined number of clusters. That is, we wanted the data to determine the number of clusters we should have. Finally, our dataset was not too big (hierarchical clustering is not suitable for extremely large datasets).

---

45 Ward, J.H. "Hierarchical Grouping to Optimize an Objective Function." *Journal of the American Statistical Association* 58(1963): 236–244.

46 Urich, D., and B. McKelvey. "General Organizational Classification: An Empirical Test Using the United States and Japanese Electronic Industry." *Organization Science* 1, no. 1 (1990): 99–118.

47 Straub, D., Boudreau, M. C., & Gefen, D. (2004). Validation guidelines for IS positivist research. Communications of the Association for Information systems, 13(1), 24.

48 http://www.socialresearchmethods.net/kb/convdisc.htm

49 http://www.socialresearchmethods.net/kb/reliable.php

50 Nunnally, J.C. *PsychometricTheory.* NewYork: McGraw-Hill, 1978.

51 Chin, W. W. (2010). How to write up and report PLS analyses. In *Handbook of partial least squares* (pp. 655-690). Springer, Berlin, Heidelberg.

52 http://www.statisticssolutions.com/structural-equation-modeling/

53 These methodology considerations are supported by: Chin, W.W. (1998). Issues and opinions on structural equation modeling. *MIS Quarterly*, 22(2), vii-xvi; Gefen, D., Straub, D. W., & Rigdon, E. E. (2011). An update and extension to SEM guidelines for administrative and social science research. *MIS Quarterly*, 35(2), iii-xiv.; and Hulland (1999). Hulland, J. (1999). Use of partial east squares (PLS) in strategic management research: A review of four recent studies. *Strategic Management Journal*, 20(2), 195-204.

54 http://methods.sagepub.com/reference/encyc-of-research-design/n77.xml

- **Measurement model**. Prior to conducting analysis, constructs were identified using exploratory factor analysis with principal component analysis using varimax rotation.[47] Statistical tests for convergent and divergent validity[48] and reliability[49] were confirmed using average variance extracted (AVE), correlation, cronbach's alpha,[50] and composite reliability.[51] The constructs passed these tests, therefore exhibiting good psychometric properties.

- **Structural equation modeling**. The structural equation models (SEM)[52] were tested using Partial Least Squares (PLS) analysis, which is a correlation-based SEM. We utilize PLS for our analysis for several reasons: It does not require assumptions of normality in the data, it is well suited to exploratory and incremental research, and the analysis optimizes for prediction of the dependent variable (vs testing for model fit of the data).[53] SmartPLS 3.2.8 was used. When controlling for industry,[54] no significant effect was found except for retail (at $p < 0.05$ level), as noted in the text. When controlling for enterprise (organizations with 5,000 or more employees), a significant effect was found (where $p < 0.001$). When controlling for years of experience, no significant effect was found. All paths shown in the SEM figures are $p < .001$, except the following, which are $p < 0.05$: Continuous delivery → Burnout, Change approvals → Software delivery performance, Continuous integration → Continuous delivery, Culture → Software delivery performance (in the main model), and External search → Productivity, Monitoring → Technical debt, and Code maintainability → Technical debt (in the productivity model).

ACKNOWLEDGEMENTS

Our research is informed by our work and conversations with the DevOps, Agile, and broader technical community; many thanks to all of our peers and colleagues who so openly share their experiences, stories, success, challenges, and failures. We use these to guide our literature review and shape our research questions each year.

The authors would like to thank several people for their input and guidance on the report this year. All acknowledgements are listed alphabetically by type of contribution.

Thanks to **Adrian Cockroft**, **Sam Guckenheimer**, **Gene Kim**, and **Kelsey Hightower** for overall guidance on the report; your input as advisors helped highlight key research ideas and direction.

For research advice, topic review, and input on measurement items, we thank the **Engineering Productivity Research** team at Google and in particular **Collin Green**, **Ciera Jaspan**, **Andrea Knight-Dolan**, and **Emerson Murphy**-**Hill**. Special thanks to the **Project Aristotle** team for their insights on psychological safety and for sharing their research instrument.

Our work is not possible without the careful review of subject matter experts; for their time on supplementary topic review and input on measurement items, we are extremely grateful.

Many thanks to **Angie Jones**, **Ashley McNamara**, **Betsy Beyer**, **Bridget Kromhout**, **Courtney Kissler**, **J. Paul Reed**, **Mike McGarr**, **Nathen Harvey**, **Seth Vargo**, and **Xavier Velasquez**.

Special thanks to **David Huh** for providing analysis support for this year's survey and report.

We thank our detailed technical readers who offered feedback and helped us refine the report: **Nathen Harvey**, **Tom Limoncelli**, **Caitie McCaffrey**, **Rachel Potvin**, **Corey Quinn**, and **Xavier Velasquez**.

The authors would like to thank **Cheryl Coupé** for her careful eye and meticulous work editing this year's report.

Report layout and design by **Siobhán Doyle**.
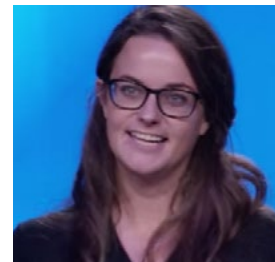
AUTHORS

Dr. Nicole Forsgren leads the DORA team, now with Google Cloud. She has led the largest studies of DevOps to date, as the principal investigator on the State of DevOps Reports for the past six years and lead author on the Shingo Publication Award-winning book *Accelerate: The Science of Lean Software and DevOps.* She has been a professor, sysadmin, and performance engineer. Nicole's work has been published in several peer-reviewed journals. Nicole earned her PhD in Management Information Systems from the University of Arizona.

Dr. Dustin Smith is a human factors psychologist and senior user experience researcher at Google. He has studied how people are affected by the systems and environments around them in a variety of contexts: software engineering, free-to-play gaming, healthcare, and military. His research at Google has emphasized identifying areas where software developers can feel happier and more productive during development. Dustin received his PhD in Human Factors Psychology from Wichita State University.

Jez Humble Jez Humble is co-author of several books on software including Shingo Publication Award winner *Accelerate, The DevOps Handbook, Lean Enterprise,* and the Jolt Award-winning *Continuous Delivery.* He has spent his career tinkering with code, infrastructure, and product development in companies of varying sizes across three continents. He works for Google Cloud as a technology advocate, and teaches at *UC Berkeley*.
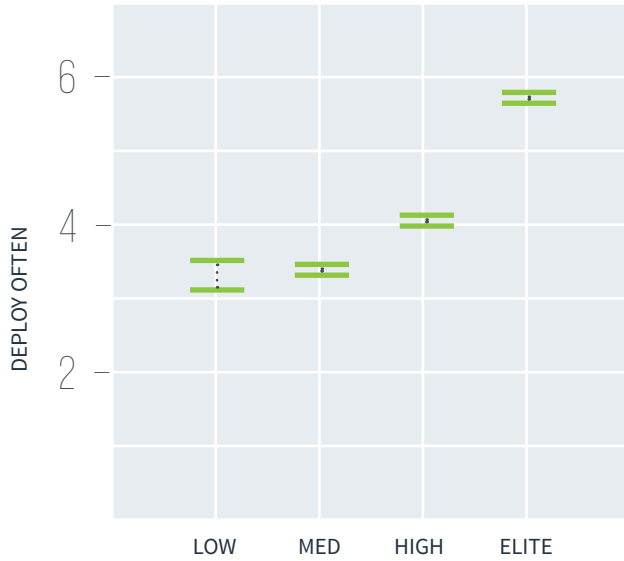
Jessie Frazelle is an independent consultant. She's been an engineer at various startups as well as Google, Microsoft, and GitHub. She's observed a lot of different development and infrastructure practices by working on the tools themselves, including Docker and Kubernetes, and by also being an end user of various PaaS. She likes to see things from all perspectives, jumping back and forth from developing tools to using them in production.
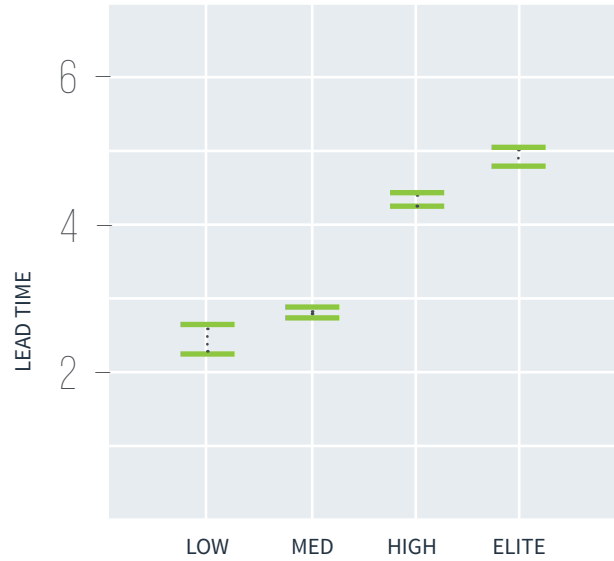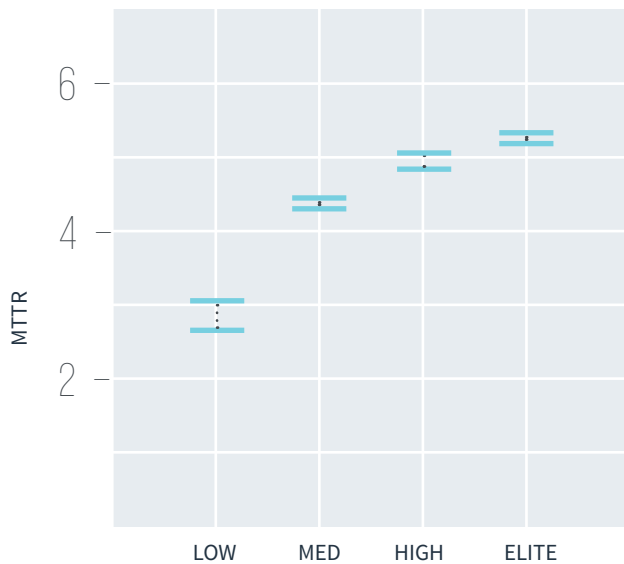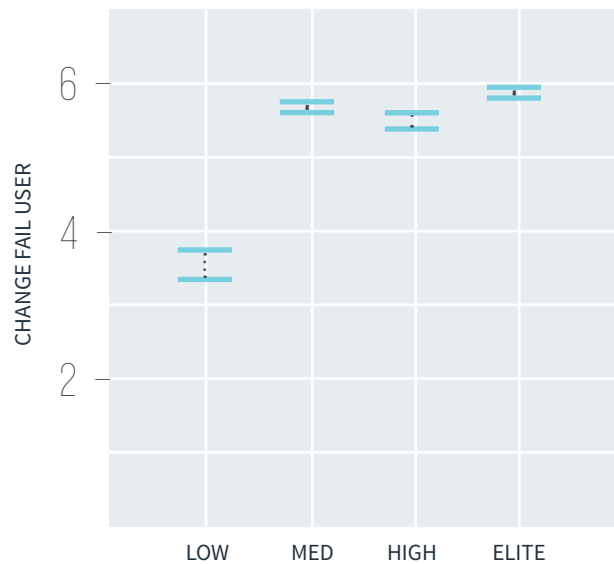
APPENDIX A

DEPLOY FREQUENCY

LEAD TIME FOR CHANGES

TIME TO RESTORE SERVICE

CHANGE FAIL RATE

DEPLOY FREQUENCY:

1 = Fewer than once per six months

2 = Between once per month
and once every 6 months

3 = Between once per week and once per month

4 = Between once per day and once per week

5 = Between once per hour and once per day

6 = On demand (multiple deploys per day)

LEAD TIME FOR CHANGES

1 = More than six months

2 = Between one month and six months

3 = Between one week and one month

4 = Between one day and one week

5 = Less than one day

6 = Less than one hour

TIME TO RESTORE SERVICE

1 = More than six months

2 = Between one month and six months

3 = Between one week and one month

4 = Between one day and one week

5 = Less than one day

6 = Less than one hour

CHANGE FAIL RATE

1 = 76%-100%

2 = 61%-75%

3 = 46%-60%

4 = 31%-45%

5 = 16%-30%

6 = 0%-15%

APPENDIX B

Strategies for Scaling DevOps

- **Training Center (sometimes referred to as a DOJO)** - Where people are taken out of their normal work routines to learn new tools or technologies, practices, and even culture for a period of time, and then put back into their normal work environment with the goal (hope?) that their new way of working will stick and possibly even spread out to others.

- **Center of Excellenc**e - Where all expertise lives and then consults out to others.

- **Proof of Concept but Stall** - A Proof of Concept (PoC) project, where a central team is given the freedom to build in whatever way they feel is best, often by breaking organizational norms (and often formal rules). However, the effort stalls after the PoC.

- **Proof of Concept as a Template** - Starting with a small Proof of Concept (PoC) project (described above), and then replicating this pattern in other groups, using the first as a pattern.

- **Proof of Concept as a Seed** - Starting with a small Proof of Concept (PoC), then spreading PoC knowledge to other groups. This is done by breaking up PoC (either the first PoC group or subsequent/ parallel PoC groups) and sending them to other groups as a way to share the knowledge and practices learned. This may also be described as a rotation, where the PoC members are immersed in other teams to spread the new practices and culture and used as teachers. They may stay in this new group indefinitely or just long enough to ensure the new practices are sustainable.

- **Communities of Practice** - Where groups that share common interests in tooling, language, or methodologies are fostered within an organization to share knowledge and expertise with each other, across teams, and around the organization.

- **Big Bang** - Where the whole organization transforms to DevOps methodologies (however they choose to define it) all at once, often with top-down directive.

- **Bottom-up or Grassroots** - Where small teams close to the work pull together resources to transform and then informally share their success throughout the organization and scale without any formal organizational support or resources.

- **Mashup** - Where the org implements several approaches described above, often only partially executed or with insufficient resources or prioritization to enable success.