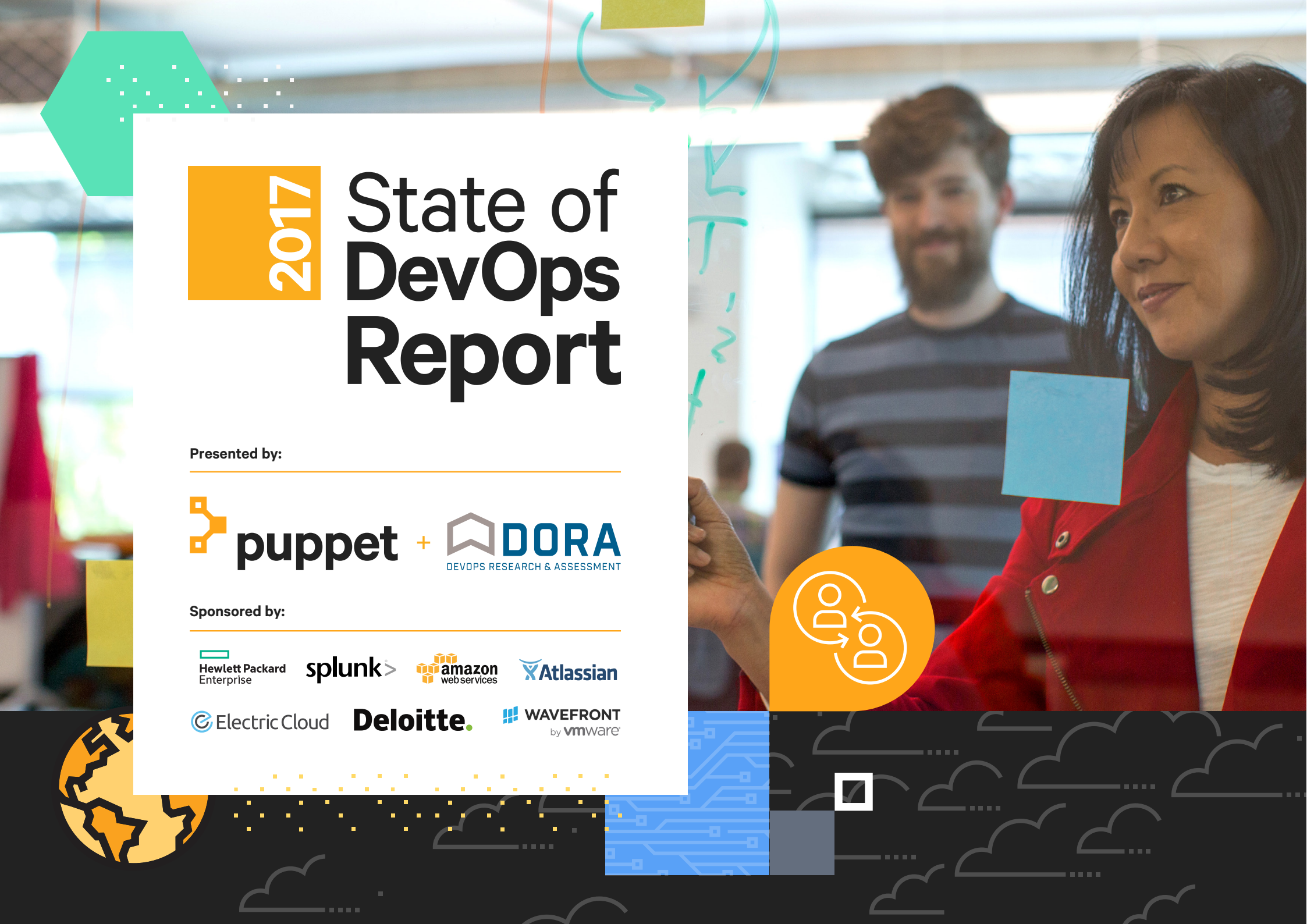


2017 State of DevOps Report

Presented by:

puppet + DORA DEVOPS RESEARCH & ASSESSMENT

Sponsored by:

Hewlett Packard Enterprise
splunk>
amazon web services
Atlassian
Electric Cloud
Deloitte.
WAVEFRONT by vmware

Contents

Back to Contents

Contents

Executive summary

Today, DevOps is an understood set of practices and cultural values that has been proven to help organizations of all sizes improve their software release cycles, software quality, security, and ability to get rapid feedback on product development. Over the past six years and more than 27,000 DevOps survey responses, we've provided strong evidence that DevOps practices lead to higher IT performance. This higher performance delivers improved business outcomes, as measured by productivity, profitability, and market share.

This year, our research also found that the beneficial effects of DevOps go beyond financial results. All organizations, both for-profit and not-for-profit, are better able to achieve their goals, no matter what their mission is.

We looked at how effective leaders influence the technical practices and process improvement changes that lead to higher IT and organizational performance. We confirmed that automation is a key differentiator for organizations. We delved deeper into how an application's architecture, and the structure of the teams that build it, can affect an organization's ability to develop and deliver software.

We hope that the high-level and tactical findings presented in this report will help you advance the progress of your own DevOps journey.

Key findings

Transformational leaders share five common characteristics that significantly shape an organization's culture and practices, leading to high performance.

The characteristics of transformational leadership — vision, inspirational communication, intellectual stimulation, supportive leadership, and personal recognition — are highly correlated with IT performance. High-performing teams have leaders with the strongest behaviors across these dimensions. Low-performing teams reported the lowest levels of these traits. Teams that reported the least transformative leaders were half as likely to be high performers.

High-performing teams continue to achieve both faster throughput and better stability.

The gap between high and low performers narrowed for throughput measures, as low performers reported improved deployment frequency and lead time for changes, compared to last year. However, the low performers reported slower recovery times and higher failure rates. We think that pressure to deploy faster and more often causes lower performers to pay insufficient attention to building in quality.

Key findings (continued)

Automation is a huge boon to organizations.

High performers automate significantly more of their configuration management, testing, deployments and change approval processes than other teams. The result is more time for innovation and a faster feedback cycle.

DevOps applies to all organizations.

This year we looked at both financial and non-financial measures of organizational performance. We found that high performers were twice as likely to achieve their own reported goals across both financial and non-financial measures. Last year's research showed that whether you're deploying commercial off-the-shelf software (COTS) or microservices in the cloud, you can attain high performance with DevOps practices. This year, we've included guidance for how to rethink COTS in a DevOps world.

Loosely coupled architectures and teams are the strongest predictor of continuous delivery.

If you want to achieve higher IT performance, start shifting to loosely coupled services — services that can be developed and released independently of each other — and loosely coupled teams, which are empowered to make changes. This shift will demand significant investment for those enterprises that require many handoffs and approvals to get work from the drawing board into production. The benefit of loosely coupled teams and services: higher throughput *and* higher quality and stability.

Lean product management drives higher organizational performance.

Lean product management practices help teams ship features that customers actually want, more frequently. This faster delivery cycle lets teams experiment, creating a feedback loop with customers. The result? The entire organization benefits, as measured by profitability, productivity, and market share.

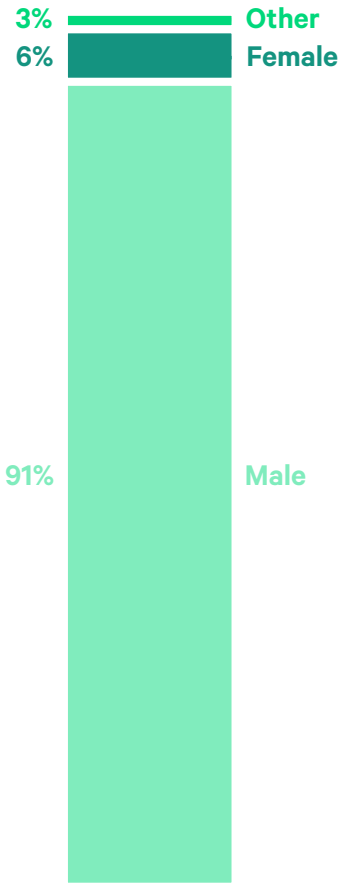
Demographics & firmographics

We've analyzed more than 27,000 survey responses from IT professionals, developers and executives over the past six years, providing the most comprehensive, evolving study of DevOps as it is practiced today. This year, 3,200 people around the world participated in the survey.

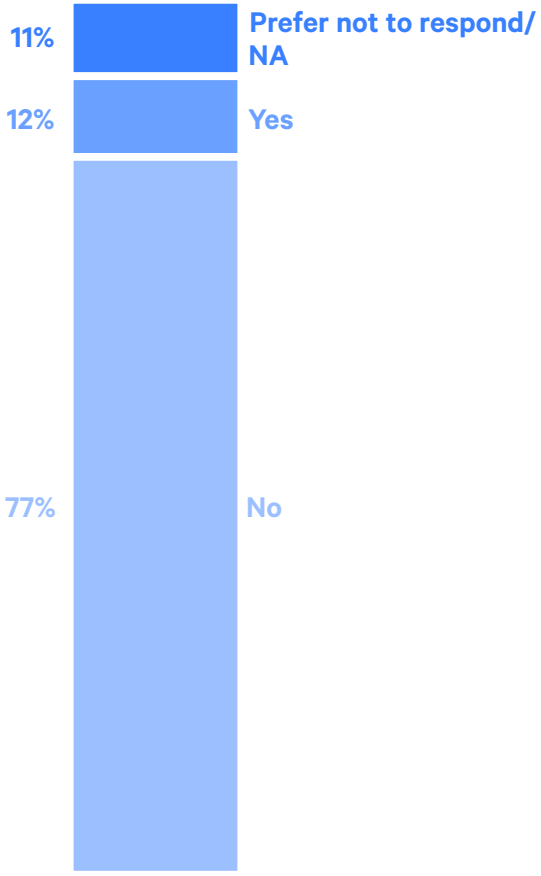
As DevOps evolves and spreads, we've noticed that the percentage of people working on DevOps teams increases each year. In 2014, 16 percent of our respondents worked on DevOps teams. Three years later, 27 percent of respondents work on such a team. We feel this increase represents both an acknowledgement that DevOps works, and the fact that DevOps teams represent a strategy for shifting the entire organization from older ways of working to newer DevOps processes.

Demographics

Gender

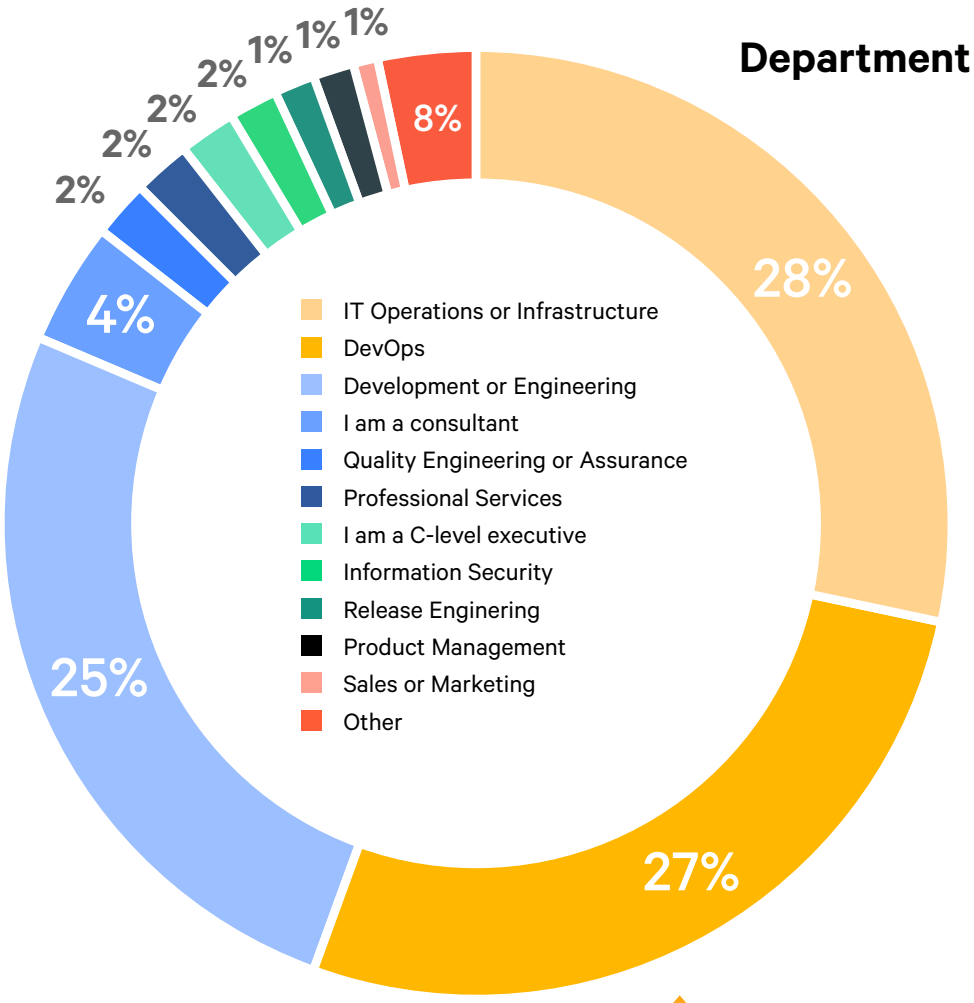


Member of underrepresented group*



* *Note: This year we asked survey respondents if they identify as a member of an underrepresented group, and let the respondent define that for themselves. It could refer to their race, gender or other characteristics they feel make them underrepresented.*

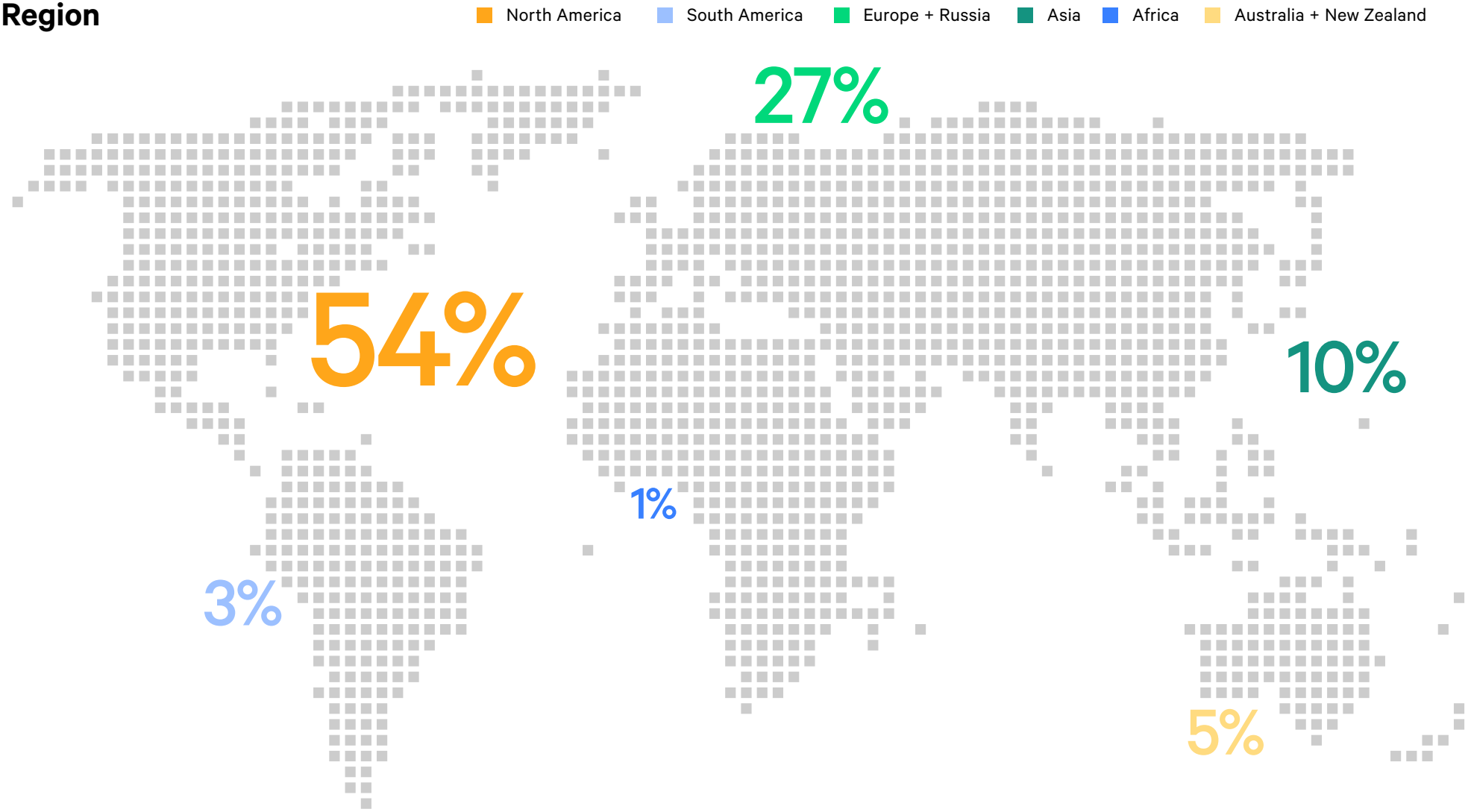
Department



DevOps teams increased from 16% in 2014 to 19% in 2015 to 22% in 2016 to 27% in 2017.

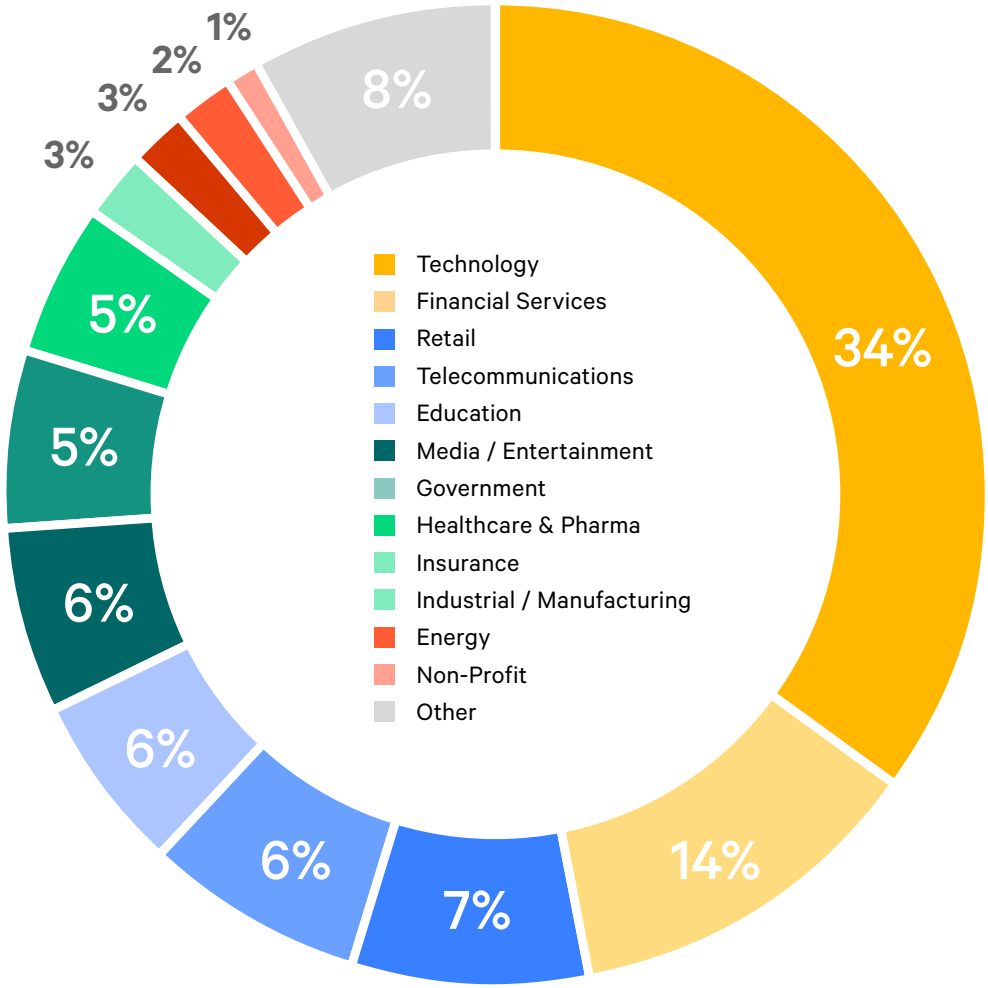
Demographics

Region

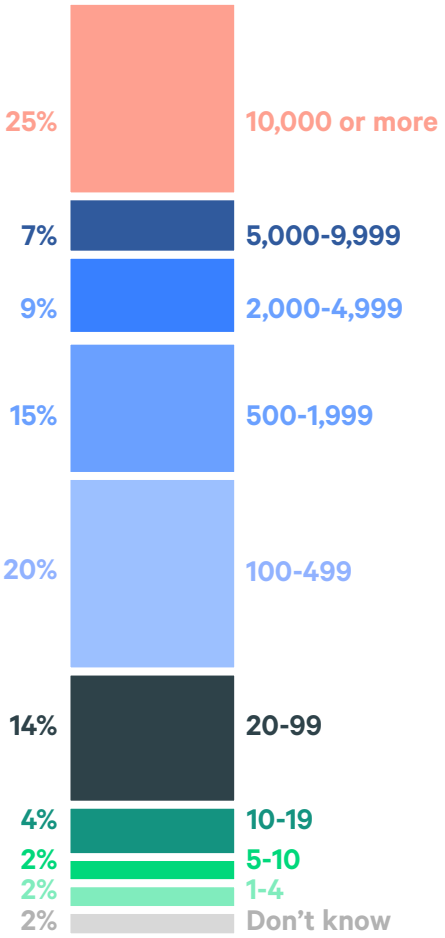


Demographics

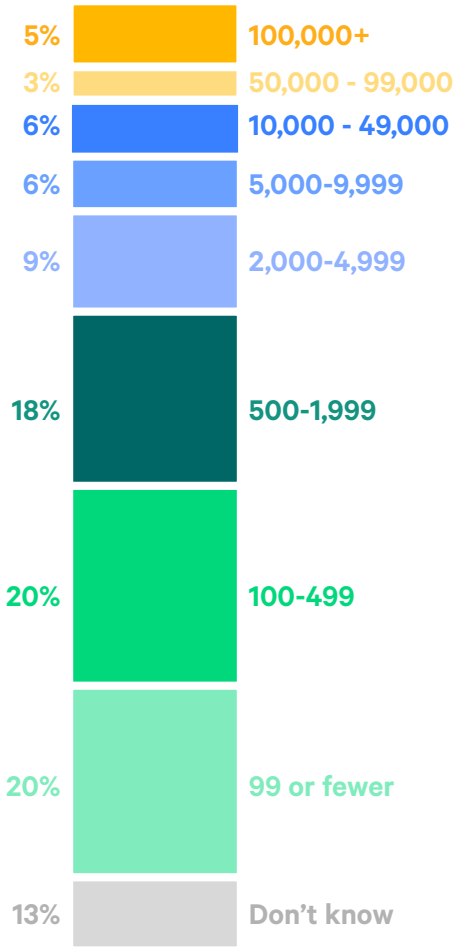
Industry

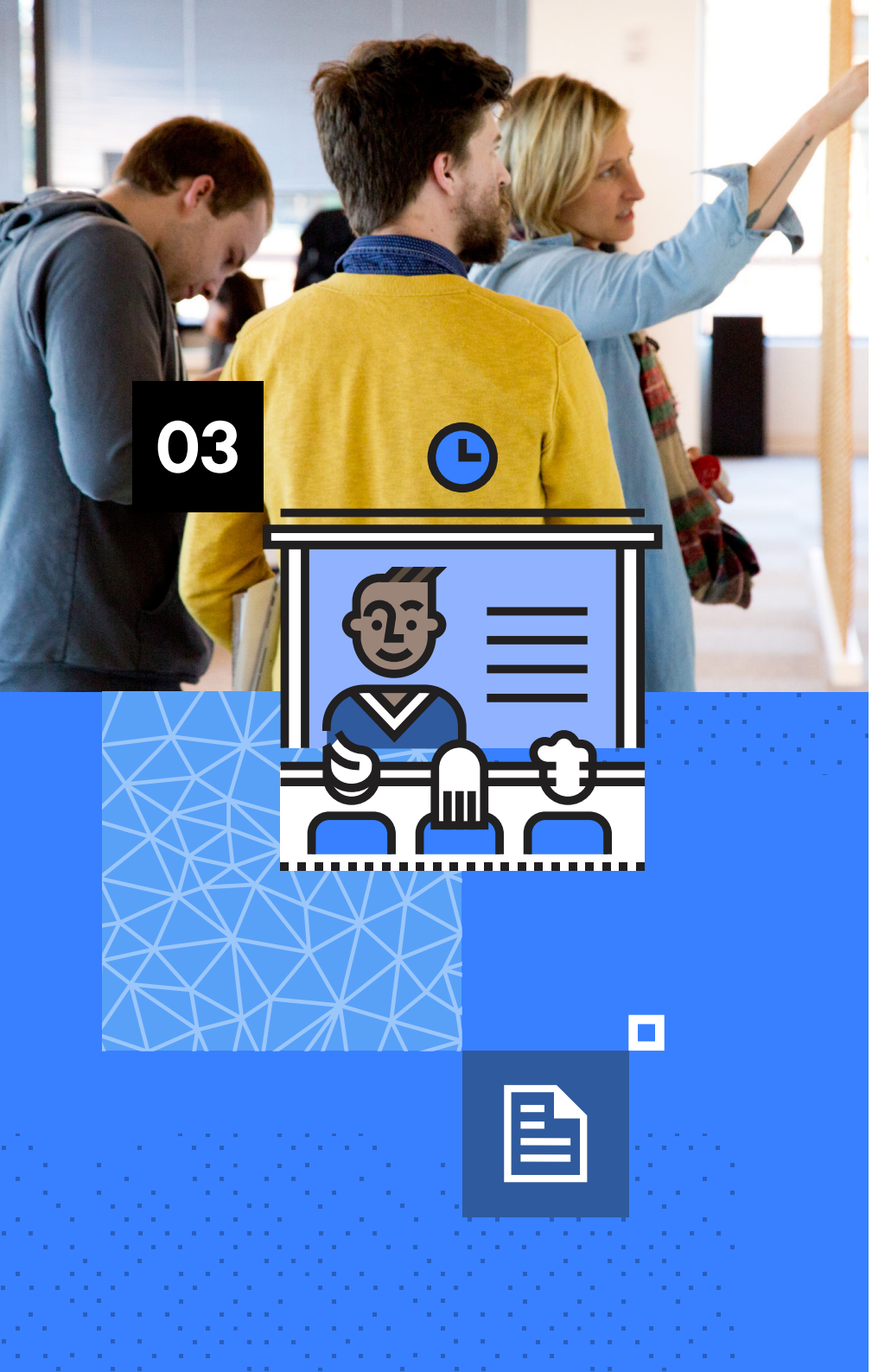


Number of employees



Number of servers





Transformational leadership

Not sure of how important IT leadership is? Consider this: By 2020, half of the CIOs who have not transformed their teams' capabilities will be displaced from their organizations' digital leadership teams.[1]

That's because leadership really does have a powerful impact on results. A good leader affects a team's ability to deliver code, architect good systems, and apply lean principles to how the team manages its work and develops products. All these things then have a measurable impact on an organization's profitability, productivity, and market share. For nonprofits and government organizations that don't think about profitability, leadership's impact can be seen in customer satisfaction, efficiency and the ability to achieve organizational goals.

---

1 Gartner Predicts - IT Infrastructure & Operations. (link to infographic on gartner.com)

One of the exciting research focus areas this year is investigating the leadership characteristics that help drive high performance. In our opinion, this has been one of the more overlooked topics in DevOps, despite the fact that transformational leadership is essential for:

- Establishing and supporting generative and high-trust cultural norms.
- Implementing technologies and processes that enable developer productivity, reducing code deployment lead times and supporting more reliable infrastructures.
- Supporting team experimentation and innovation, to create and implement better products faster.
- Working across organizational silos to achieve strategic alignment.

Within the DevOps community, we have sometimes been guilty of maligning leadership — for example, when middle managers or conservative holdouts prevent us from making changes needed to improve IT and organizational performance.

Still, one of the most common questions we hear is, "How do we get leaders on board, so we can make the necessary changes?" Everyone recognizes that engaged leadership is essential for successful DevOps transformations. Leaders have the authority and budget to make the large-scale changes often needed; to provide visible support when a transformation is underway; and to change the incentives of entire groups of engineers, whether they are in development, QA, operations, or information security. Leaders are the ones who set the tone of the organization, and reinforce the desired cultural norms.

For this year's study, we used a measure of transformational leadership that includes five dimensions (Rafferty and Griffin 2004).[2] According to this model, the five characteristics of a transformational leader are:

- **Vision.** Has a clear concept of where the organization is going and where it should be in five years.
- **Inspirational communication.** Communicates in a way that inspires and motivates, even in an uncertain or changing environment.
- **Intellectual stimulation.** Challenges followers to think about problems in new ways.
- **Supportive leadership.** Demonstrates care and consideration of followers' personal needs and feelings.
- **Personal recognition.** Praises and acknowledges achievement of goals and improvements in work quality; personally compliments others when they do outstanding work.

2 Rafferty, A. E., & Griffin, M. A. (2004). *Dimensions of transformational leadership: Conceptual and empirical extensions.* The Leadership Quarterly, 15(3), 329-354.

What is transformational leadership?

Transformational leadership is a model in which leaders inspire and motivate followers to achieve higher performance by appealing to their values and sense of purpose, facilitating wide-scale organizational change. These leaders encourage their teams to work towards a common goal through their vision, values, communication, example-setting, and their evident caring about their followers' personal needs.

It has been observed that there are similarities between servant leadership and transformational leadership, but they differ in the leader's focus. Servant leaders focus on their followers' development and performance, whereas transformational leaders focus on getting followers to identify with the organization and engage in support of organizational objectives.

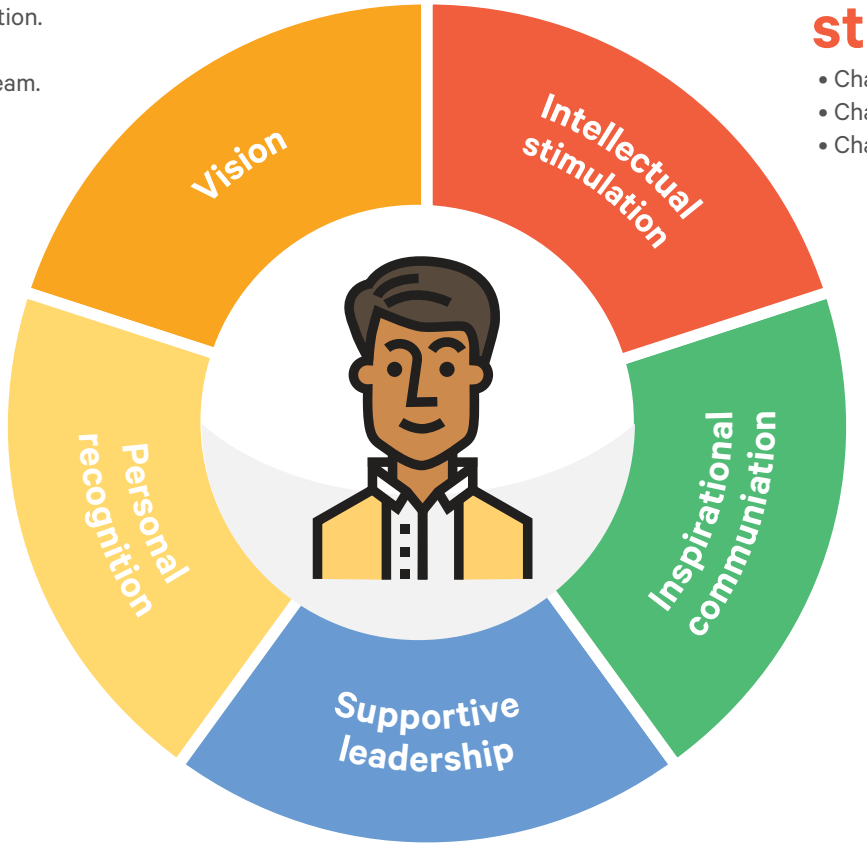
Dimensions of transformational leadership

Vision

- Understands organizational direction.
- Understands team direction.
- Understands 5-year horizon for team.

Intellectual stimulation

- Challenges team status quo.
- Challenges team to constantly ask new questions.
- Challenges team on basic assumptions about the work.



Personal recognition

- Commends team for better-than-average work.
- Acknowledges improvement in quality of work.
- Personally compliments individuals' outstanding work.

Inspirational communication

- Inspires pride in being part of the team.
- Says positive things about the team.
- Inspires passion and motivation; encourages people to see that change brings opportunities.

Supportive leadership

- Considers others' personal feelings before acting.
- Is thoughtful of others' personal needs.
- Cares about individuals' interests.

The characteristics of transformational leadership are highly correlated with IT performance. In fact, we observed significant differences in leadership characteristics between high-, medium- and low-performing IT teams. High-performing teams reported having leaders with the strongest behaviors across all dimensions: vision, inspirational communication, intellectual stimulation, supportive leadership, and personal recognition. In contrast, low-performing teams reported the lowest levels of these leadership characteristics. The differences we found were all at statistically significant levels.

What was most striking, however, was that teams with the least transformative leaders (the bottom third) were also far less likely to be high IT performers — in fact, they were half as likely to exhibit high IT performance. This validates our common experience: Though we often hear stories of DevOps and technology-transformation success coming from the grassroots, it is far easier to achieve success when you have effective leaders lending their support.

Our analysis also found that transformational leadership is highly correlated with employee **Net Promoter Score (NPS)**. We find transformational leaders in places where employees are happy, loyal, and engaged. Based on these results, and the findings in previous years' research, we expect that teams with strong transformational leaders will also prove to have generative organizational cultures (as measured by sociologist Ron Westrum[3]) and report strongly identifying with their work — constructs we measured in 2016, but that weren't included in this year's study.

A transformational leader's influence is seen in their support of their teams' work, both in technical practices and in the teams' product management capabilities. The positive (or negative) influence of leadership flows all the way through to IT performance and organizational performance.

3 Westrum, R. A typology of organisational cultures. qualitysafety.bmj.com/content/13/suppl_2/ii22.

Interestingly, we found evidence that the presence of leaders with transformational characteristics is not enough to achieve high DevOps outcomes. Of the teams that reported on how well their leaders exemplified transformational leadership characteristics, we focused on those teams whose leaders were in the top 10 percent. We thought these teams would, as a group, be the very highest performers. That was not the case — they displayed variation in performance level. This told us that transformational leadership behavior is not enough, by itself, to drive high IT performance.

Why was this the case? Because leaders cannot achieve DevOps outcomes on their own. DevOps success also depends on a suitable architecture, good technical practices, use of lean management principles, and all the other factors that we've studied over the years.

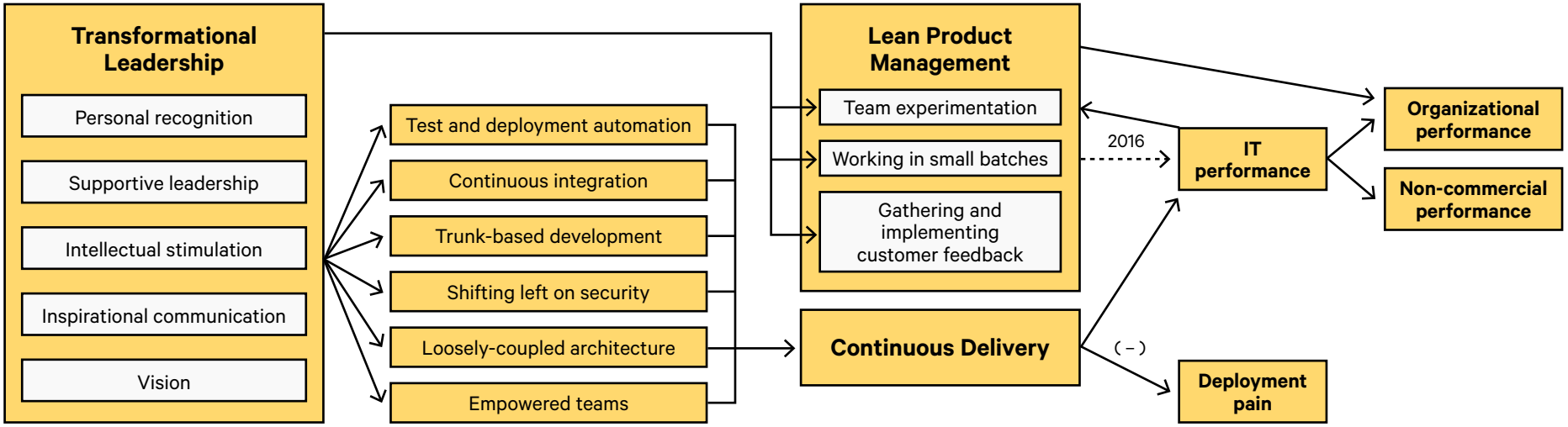
In summary, we found that good leaders help build great teams, great technology, and great organizations *indirectly*, by enabling teams to re-architect their systems and implement continuous delivery and lean management practices. Transformational leadership enables the necessary practices that correlate with high performance, and it also supports effective communication and collaboration between team members in pursuit of organizational goals. Such leadership provides the foundation for a culture in which continuous experimentation and learning is part of everybody's daily work.

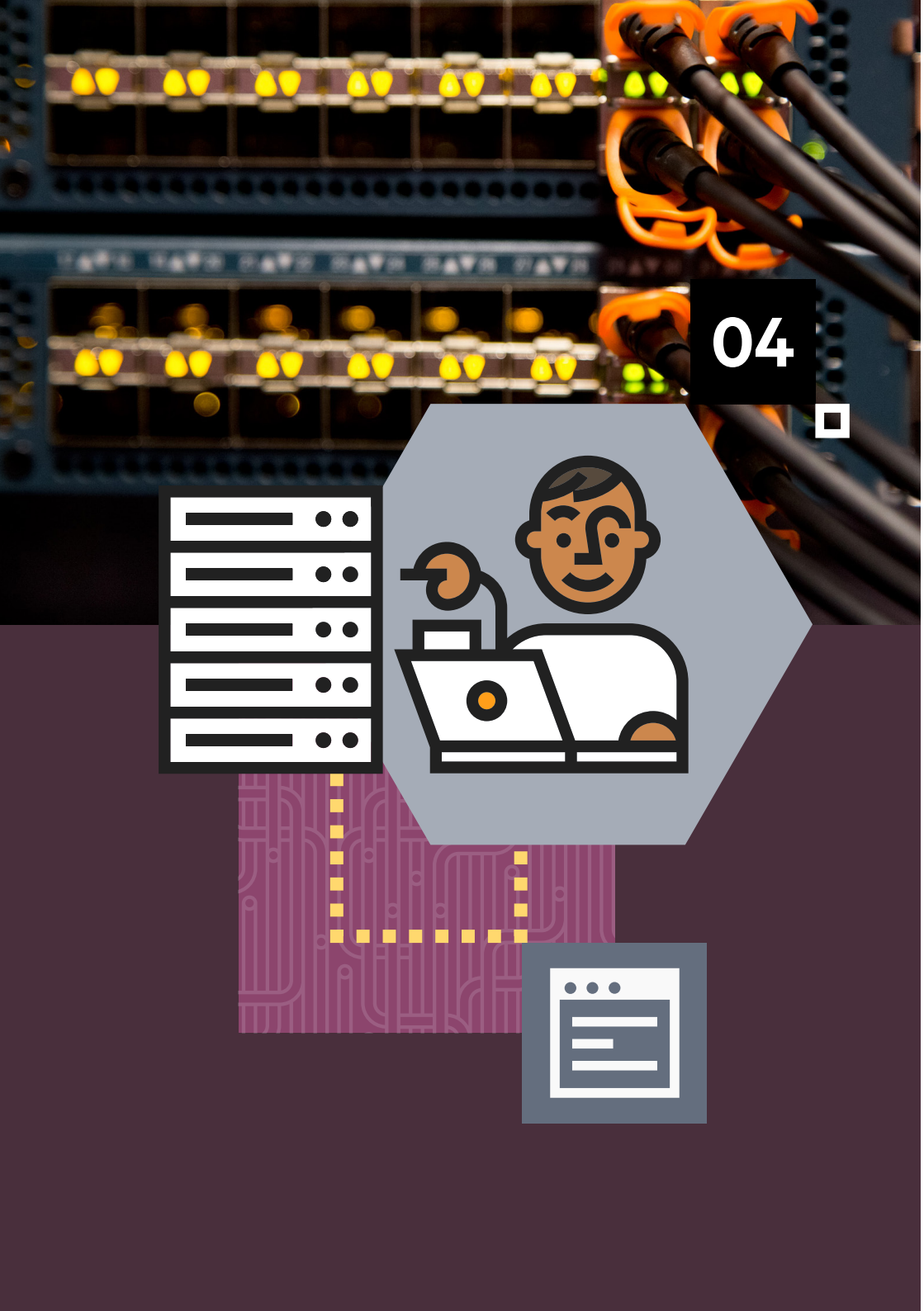
The behavior of transformational leaders thus enhances and enables the values, processes and practices that our research has identified. Transformational leadership is not a separate behavior or a new set of practices; instead, it amplifies the effectiveness of the technical and organizational practices we have been studying over several years.

The figure below is the model we tested in this year's research. It is a structured equation model (SEM), and is a predictive model used to test relationships. Each box in the figure represents a construct we measured in our research, and each arrow represents relationships between the constructs. To interpret the model, all arrows can be read using the words *predicts, affects, drives,* or *impacts*.

For example, IT performance predicts organizational performance. If you see a (-) next to one of the arrows, it means the relationship is negative: Continuous delivery negatively impacts deployment pain. All arrows in the model represent statistically significant relationships. The relationship between lean product management and IT performance was tested in 2016; this is discussed more in the **Lean product management** section.

Figure 1. Structured equation model showing relationships between constructs





IT performance & organizational performance

Today, every organization relies on software and IT to advance its purpose, whether that's growing a profitable business or creating a social benefit. Organizations look to DevOps because of the growing evidence that DevOps practices help you deliver software faster, more reliably, and with fewer errors.

We measure IT performance along two main dimensions: throughput of code and stability of systems. Throughput is measured by how frequently a team is able to deploy code and how fast it can move from committing code to deploying it. Stability is measured by how quickly the system can recover from downtime and how many changes succeed, versus how many fail.

As in previous years, we have found that high performers do significantly better than their lower-performing peers in terms of throughput and stability. In 2017, we found that the high performers have:

- 46 times more frequent code deployments
- 440 times faster lead time from commit to deploy
- 96 times faster mean time to recover from downtime
- 5 times lower change failure rate (changes are 1/5 as likely to fail)

When compared to the 2016 results, the gap between high and low performers narrowed for throughput (deployment frequency and change lead time), and widened for stability (mean time to recover and change failure rate). We speculate that this is due to low-performing teams working to increase speed, but not investing enough in building quality into the process. The result is larger failures, and more time to restore service. High performers understand that they don't have to trade speed for stability or vice versa, because by building quality in, they get both.

Table 1: Changes in IT performance of high performers, 2016 to 2017

IT performance metrics	2016	2017
Deployment frequency	200x more frequent	46x more frequent
Lead time for changes	2,555x faster	440x faster
Mean time to recover (MTTR)	24x faster	96x faster
Change failure rate	3x lower (1/3 as likely)	5x lower (1/5 as likely)

Some may wonder why these gaps in year-over-year performance are widening or narrowing. It is important to note that all these measures are relative: They compare the difference between the high and low performers. Between 2016 to 2017, the gap for frequency of code deployments narrowed: High performers are still shipping code as the business demands, while low performers went from shipping between once per month and once every six months in 2016, to shipping between once per week and once per month in 2017. Low performers in 2017 have also reduced their lead time for changes: from between one month and six months in 2016 to between one week and one month. This change does not mean that high performers are no longer performing as well. It simply means that low performers are doing better with throughput than they were, on average, and we applaud them for this improvement.

In contrast, high performers have gained an even greater advantage over the past year when it comes to recovering from production and infrastructure outages, and preventing failures in the first place. This is likely giving them an advantage in delighting their customers, because they have many more chances to deliver new value, and what they release is of higher quality. The result is faster time to market, better customer experience, and higher responsiveness to market changes.

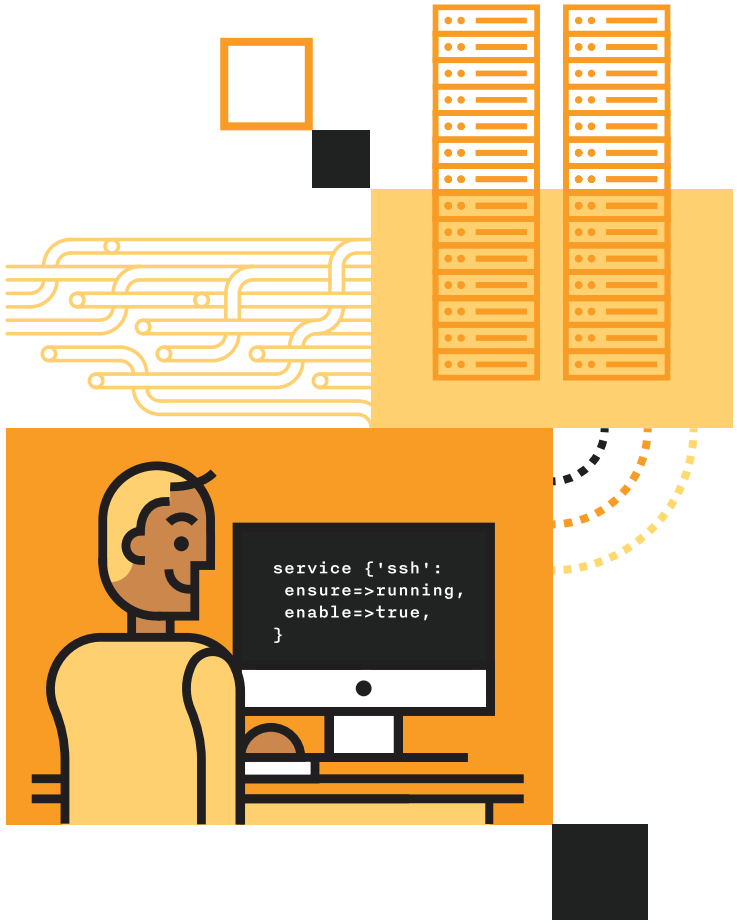


Table 2: 2017 IT performance by cluster

Survey questions	High IT performers	Medium IT performers	Low IT performers
Deployment frequency *For the primary application or service you work on, how often does your organization deploy code?*	On demand (multiple deploys per day)	Between once per week and once per month	Between once per week and once per month*
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 commit to code successfully running in production)?*	Less than one hour	Between one week and one month	Between one week and one month*
Mean time to recover (MTTR) *For the primary application or service you work on, how long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?*	Less than one hour	Less than one day	Between one day and one week
Change failure rate *For the primary application or service you work on, what percentage of changes results either in degraded service or subsequently requires remediation (e.g., leads to service impairment, service outage, requires a hotfix, rollback, fix forward, patch)?*	0-15%	0-15%	31-45%

* *Note: Low performers were lower on average (at a statistically significant level), but had the same median as the medium performers.*

For information on how we determined high, medium and low IT performance, please see the Methodology section.

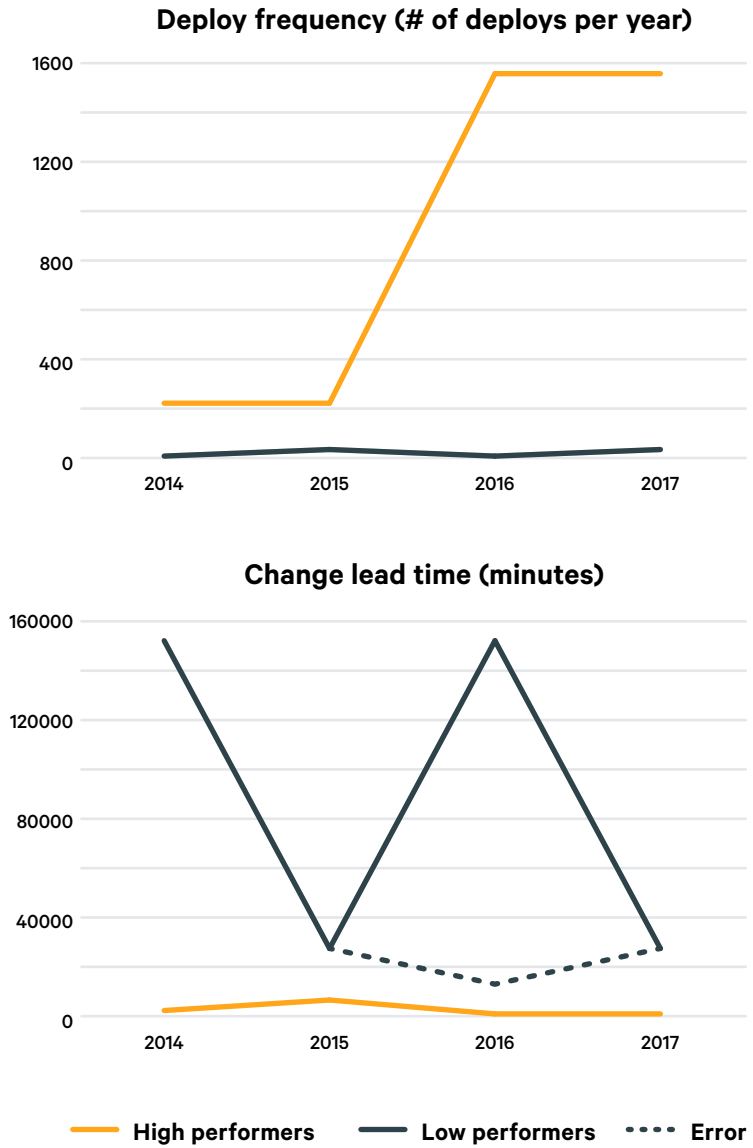
Throughput

Deployment frequency

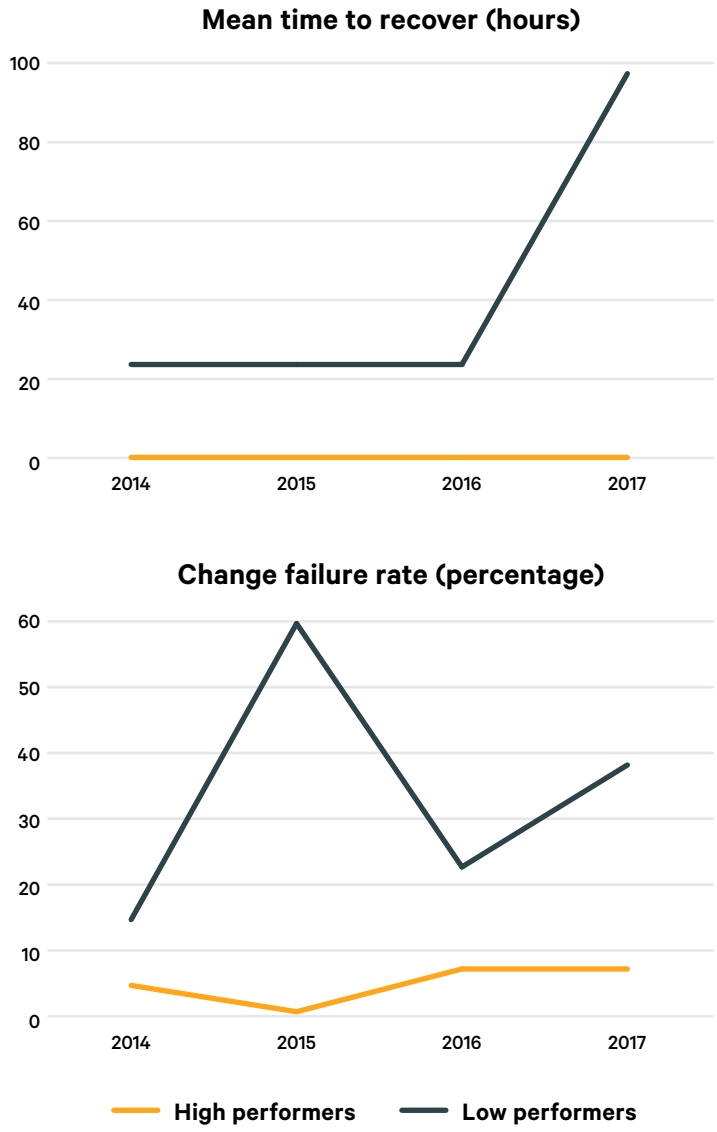
This year, high performers reported that they are routinely deploying on demand, performing multiple deployments per day. Low performers, by comparison, reported deploying between once per week and once per month; this has improved from last year. We normalized these ranges to 1,460 deploys per year (four deploys per day x 365 days) for high performers, and 32 deploys per year for low performers (mean of 52 deploys and 12 deploys). Working with these normalized figures shows that high performers deploy code 46 times more frequently than their low-performing peers. It's worth noting that four deploys per day is conservative for companies like Etsy that report deploying 80 times per day, and for companies like Amazon and Netflix that deploy thousands of times per day (aggregated over the hundreds of services that comprise their production environments).

Change lead time

Similarly, high performers reported that their lead time required to deploy changes into production (i.e., go from code committed to code deployed and running successfully in production) was less than one hour, whereas low performers required lead times between one week and one month. So the high performers had 440 times faster change lead times than low performers. For our calculations, we used lead times of 60 minutes for high performers, and 26,940 minutes for low performers (the mean of 10,080 minutes per week and 43,800 minutes per month).



* *Note: Last year, our graph incorrectly reported the lead time for low performers, as shown by the dotted line.*



Stability

Mean time to recover (MTTR)

High performers reported that their MTTR was less than one hour, while low performers reported between one day and one week. In other words, high performers had 96 times faster MTTR than low performers. For this calculation, we chose conservative time ranges: one hour for high performers, and the mean of one day (24 hours) and one week (168 hours) for low performers. As already noted, MTTR worsened for low performers, compared to last year.

Change failure rate

High performers reported a change failure rate between zero and 15 percent, while low performers reported change failure rates of 31-45 percent. We took the means of these two ranges, giving us a 7.5 percent change failure rate for high performers and 38.5 percent for low performers. This means high performers have change failure rates that are five times lower than low performers. As already noted, change failure rates worsened for low performers, compared to last year.

Note: The increase in 2016 for high performers is due to a change in how we measured change fail rate. Prior to 2016, we allowed respondents to input a number. In 2016 and 2017, we changed the answers to ranges. We took the average of the range, 0-15 percent, which gives us 7.5 percent.

Automation

Last year, as a proxy for quality, we analyzed the amount of time teams spent on rework and unplanned work. This year, in addition to looking at rework, we also looked at how much of their work each of our performance groups was still doing manually across these practices: configuration management, testing, deployment, and change approval.

We found striking results. When we compared high performers to low performers, we found that high performers are doing significantly less manual work, and so have automated:

- 33 percent more of their configuration management.
- 27 percent more of their testing.
- 30 percent more of their deployments.
- 27 percent more of their change approval processes.

Automation is a huge boon to organizations. With more work automated, high performers free their technical staff to do innovative work that adds real value to their organizations. A good example of this can be seen in the transformation at HP LaserJet.[4] The firmware division was on the critical path for hardware releases; by undertaking a continuous improvement initiative and investing in automation — including significant investments in automated testing — HP LaserJet was able to increase time spent on developing new features by ***700 percent***.

What we've also learned, in our work with teams and in conducting research, is that people are not very good at estimating how automated they are. What they *are* good at is estimating the percentage of their work that is still done manually. That's not surprising: Manual work is painful, so people are highly aware of it. Once work is automated, it's no longer painful, and tends to disappear from people's attention.

---

4 *The Amazing DevOps Transformation of the HP LaserJet Firmware Team (Gary Gruver).* itrevolution.com/the-amazing-devops-transformation-of-the-hp-laserjet-firmware-team-gary-gruver

Below you can see our findings on manual work, and how much less of it the high performers do.

Not surprisingly, high IT performers report the lowest amount of manual work across all practices — and therefore, the highest amount of automation — at statistically significant levels. (Although you may notice that low performers are slightly less manual in their configuration management and testing than medium performers, the difference is not statistically significant, and should therefore be ignored.)

One thing initially surprised us, though: Medium performers do more manual work than low performers when it comes to deployment and change approval processes, and these differences *are* statistically significant. This was particularly interesting because last year, we encountered another seemingly anomalous finding: Medium performers actually spent *more* time on rework than low performers, though they deployed more frequently. How could that be?

Table 3: Percentage of work that is done manually, by performance group.
All percentages significantly different among High, Medium, and Low IT performers, except where otherwise noted.

	High performers	Medium performers	Low performers
Configuration management	28%	47%[a]	46%[a]
Testing	35%	51%[b]	49%[b]
Deployments	26%	47%	43%
Change approval processes	48%	67%	59%

[a, b] *Not significantly different*

Based on our anecdotal experience with real teams that are accelerating on their automation journey, we now regard this phenomenon of manual work and rework as a temporary stage. At this phase of their DevOps journey, medium performers have automated enough of their work to see real results. But they have also discovered a mountain of technical debt — and that technical debt is holding them back. So the medium performers use the time freed up by increased automation to burn down the technical debt that blocks their further progress. And the ramifications of addressing that technical debt can cause teams to institute more manual controls around changes, adding layers of process that ultimately slow them down.

While the desire to add more manual controls is understandable, we urge organizations to resist this temptation. Our guidance to teams at this stage is to instead begin shifting the change review process to an earlier phase of the development cycle, and to rely on peer review and automated testing rather than a change review board. Ultimately, this will eliminate the need for a change review board altogether, and the team can move forward with a faster (and more reliable) process for reviewing changes.

Once the stage of burning down technical debt has passed, further automation becomes much more achievable, and the team can move to the next steps of its automation and DevOps journey.

Additional measures of organizational performance

For years, some people have argued that DevOps is just for unicorns — for high-profile technology companies like Google, Amazon, Netflix, and others that can hire phalanxes of engineers to work on processes, not just on the core product. We've largely moved past that perception, and leaders of mainstream enterprise companies now recognize the competitive edge that DevOps can give them. But there's still a perception that DevOps matters more to for-profit enterprises than to not-for-profit or government organizations.

Our findings this year show that the ability to develop and deliver software efficiently and accurately is a key differentiator and value driver for all organizations — for-profit, not-for-profit, educational and government organizations alike. If you want to deliver value, no matter how you measure it, DevOps is the way to go.

In 2014, we first published our findings on how IT performance predicts organizational performance. We found that high performers were twice as likely to exceed their own goals for profitability, market share and productivity.[5] One of the most frequent questions we've been asked is how this applies to not-for-profit organizations, such as military services and government agencies, universities, and non-profits.

So this year, we explored how IT performance affects an organization's ability to achieve broader organizational goals — that is, goals beyond profit and revenue measures. Because whether you're trying to generate profits or not, every organization today depends on technology to achieve its mission and provide value quickly, reliably and securely to its customers or stakeholders. Whatever the mission, how the technology organization performs has an impact on overall organizational performance.

5 Widener, S.K. (2007). An empirical analysis of the levers of control framework. Accounting, Organizations and Society, 32(7), 757-788.

We found that high performers were more than twice as likely to achieve or exceed the following objectives:[6]

- Quantity of products or services.
- Operating efficiency.
- Customer satisfaction.
- Quality of products or services provided.
- Achieving organizational and mission goals.
- Measures that demonstrate to external parties whether or not the organization is achieving intended results.

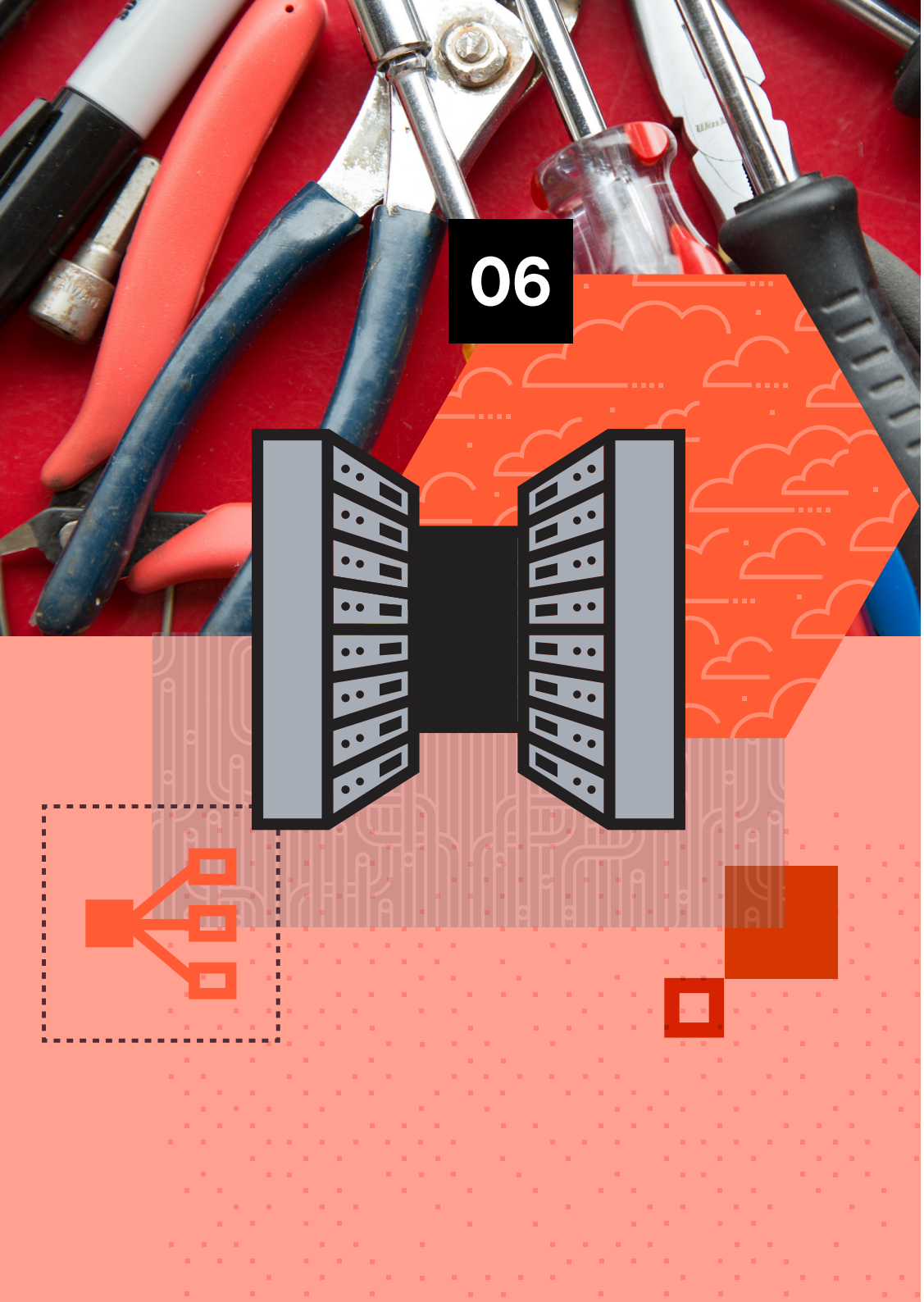
We find it very interesting that high performers in both for-profits and not-for-profits are twice as likely to achieve or exceed objectives. This suggests that DevOps enables mission achievement for *any* type of organization, independent of industry or sector. It's interesting to note that the margin of advantage for high-performing for-profit organizations — being twice as likely to achieve or exceed objectives — has remained consistent over the past four years.

DevOps in a regulated environment

For an example of how DevOps practices can be applied even in highly regulated domains, read about how 18F's open source platform-as-a-service, **cloud.gov**, significantly reduces the time and costs for taking government services live, by handling many compliance requirements for federal information systems at the platform level.

18f.gsa.gov/2017/02/02/cloud-gov-is-now-fedramp-authorized

6 Adapted from Cavalluzzo, K. S., & Ittner, C. D. (2004). Implementing performance measurement innovations: evidence from government. Accounting, Organizations and Society, 29(3), 243-267.



Technical practices

Today, DevOps is viewed as the path to faster delivery of software, greater efficiency, and the ability to pull ahead of the competition. Some, eager to start on the DevOps path, begin by researching which tools they should buy. More important than specific tools, though, are the technical practices that enable you to achieve the very things that most people turn to DevOps for.

Our research has always delved into the practices that are common to successful DevOps organizations — practices like version control, continuous integration, trunk-based development, and automation. This year, we added a focus on the structure of architecture and teams, and how they affect an organization's ability to develop and deliver software.

Continuous delivery

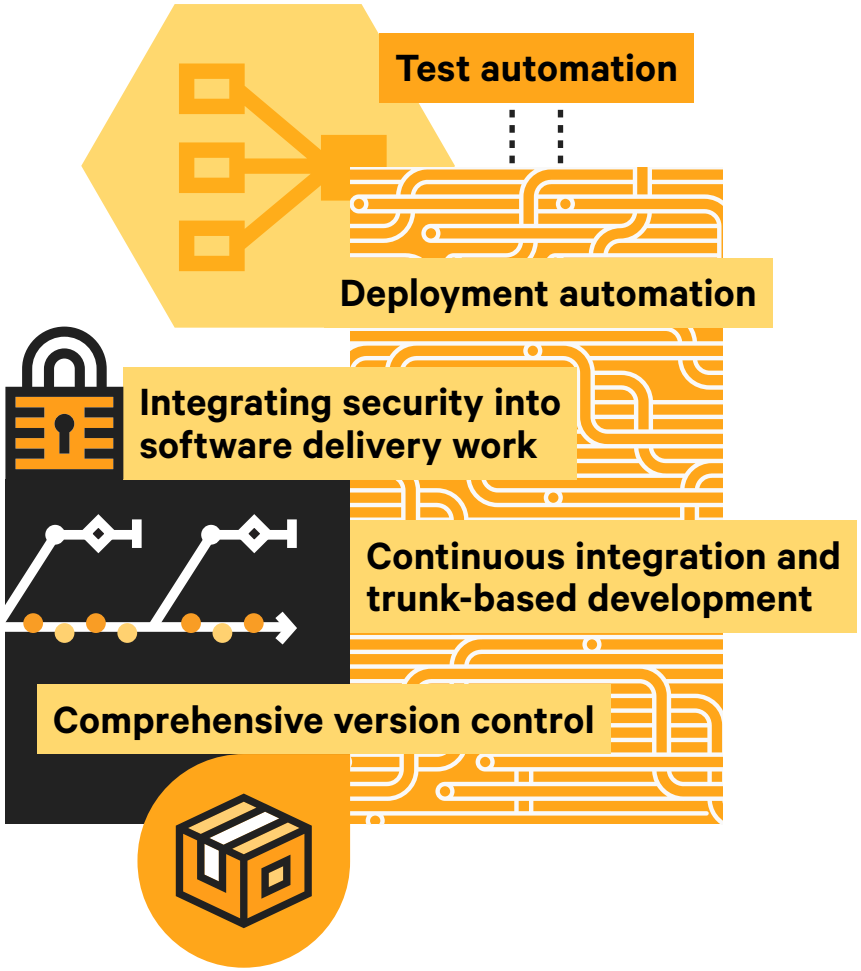
Last year, we found that the practices that make up continuous delivery — deployment automation and automated testing, continuous integration and trunk-based development, and version control for all production artifacts — have a significant predictive relationship to deployment pain, IT performance and change failure rate. In turn, IT performance predicts organizational performance, as measured by productivity, market share and profitability.

In previous years, we've modeled continuous delivery as the combination of the capabilities listed above. This year we wanted to model continuous delivery as an outcome in its own right (though we recognize it drives business outcomes such as faster time to market and higher quality). In our model, we measured continuous delivery according to a team's ability to achieve the following outcomes:

- 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 people make acting on this feedback their highest priority.

This allows us to do two interesting things. First, we can see the impact of achieving these outcomes on IT performance and deployment pain. Second, we can see which factors contribute most strongly towards achieving these outcomes, and also look at the impact of other attributes, such as having a loosely coupled architecture (defined in the Architecture section).

Factors that positively contribute to continuous delivery:



As we had hypothesized, we found that the following all positively affect continuous delivery: comprehensive use of version control; continuous integration and trunk-based development; integrating security into software delivery work; and the use of test and deployment automation. Of these, test automation is the biggest contributor.

One more interesting finding: Because this year we measured the impact of continuous delivery itself, we can say that it significantly contributes to both lower deployment pain and higher IT performance.

This year we wanted to test the impact of good architectural practices on continuous delivery. Our results confirmed the importance of architecture for achieving high performance.

Architecture

Over the past few years, we've investigated how architecture correlates with continuous delivery and IT performance. This year we wanted to take a deep dive into architecture, and test some commonly held beliefs about what constitutes effective architecture. We discovered that creating architectures and teams that are loosely coupled significantly drives the ability to practice continuous delivery.

We measured coupling between services and components by capturing whether:

- Respondents could do testing without requiring an integrated environment.
- The application respondents worked on could be deployed or released independently of other applications and services that it depends on.

To account for commercial off-the-shelf (COTS) software scenarios, we also asked if respondents' environments included COTS.

In a loosely coupled architecture, it's easy to modify or replace any individual component or service without having to make corresponding changes to the services or components that depend on it. In an organizational sense, teams can be described as loosely coupled if they don't depend upon other teams to complete their work.

In 2015, we discovered that high-performing teams were more likely to have loosely coupled architectures than medium- and low-performing teams. This was true for both for the primary application or service they were working on and for the services they had to interact with. High-performing teams were using loosely coupled architectures for both new and brownfield systems, including packaged software (COTS), embedded systems, user-installed and server-side systems.

This year we extended our research to test two new hypotheses:

1. Empowered teams that make their own tool decisions and implementations contribute to better IT performance.
2. A loosely coupled, well encapsulated architecture drives IT performance.

For our first hypothesis, we discovered that teams that can decide which tools they use do better at continuous delivery. This is in contrast to teams that can use only those tools that are mandated by a central group. Teams that can choose their own tools are able to make these choices based on how they work, and the tasks they need to perform. No one knows better than practitioners what they need to be effective, so it's not surprising that practitioner tool choice helps to drive better outcomes.

Our second hypothesis was also supported, validating and extending our earlier research. In teams with strong IT and organizational performance, the architecture of the system is designed so delivery teams can test, deploy and change their systems without depending on other teams for additional work, resources, or approvals, and with less back-and-forth communication. Therefore, we describe both the architecture and the teams as being loosely coupled.

This connection between systems architecture and communication bandwidth was first discussed by computer programmer **Melvin Conway**, who said, "organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations." Our research lends support to what is sometimes called "reverse Conway's Law," which states that organizations should architect around team boundaries to ensure that teams can get their work done — from design through to deployment — without requiring high-bandwidth communication between teams.

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. The architecture should also enable the use of test doubles and virtualization to test services or components in isolation. Service-oriented architectures are supposed to enable these outcomes, as should any true microservices architecture. However, you must be very strict about these outcomes when implementing such architectures. In real life, many so-called service-oriented architectures don't permit testing and deploying services independently from each other, and so do not enable teams to achieve higher performance.

The biggest contributor to continuous delivery — bigger even than test and deployment automation — is whether a team can do all of the following:

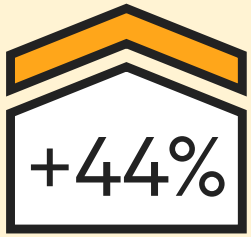
- Make large-scale changes to the design of its system without permission from someone outside the team.
- Make large-scale changes to the design of its system without depending on other teams to make changes in their own systems, or creating significant work for other teams.
- Complete its work without needing fine-grained communication and coordination with people outside the team. For example, not having to check 12 Google calendars to get feedback.
- Deploy and release its product or service on demand, independently of other services the product or service depends upon.
- Do most of its testing on demand, without requiring an integrated test environment.
- Perform deployments during normal business hours with negligible downtime.

Quality and security

Last year we introduced a new construct for quality so we could measure the impact of continuous delivery practices on software quality. Because quality is inherently difficult to measure, we used unplanned work and rework as proxies. We found that high-performing organizations spend 22 percent less time on unplanned work and rework. As a result, they are able to spend 29 percent more time on new work, such as new features or code. This year, we are happy to report that high-performing organizations spend **21 percent less time on unplanned work and rework**, and **44 percent more time on new work**.

Last year, we also found that high performers spend 50 percent less time remediating security issues than low performers. We were able to validate that again this year. These results point to the need to involve security and quality teams in the development process early and often.

less time spent on planned work and rework



more time spent on new work

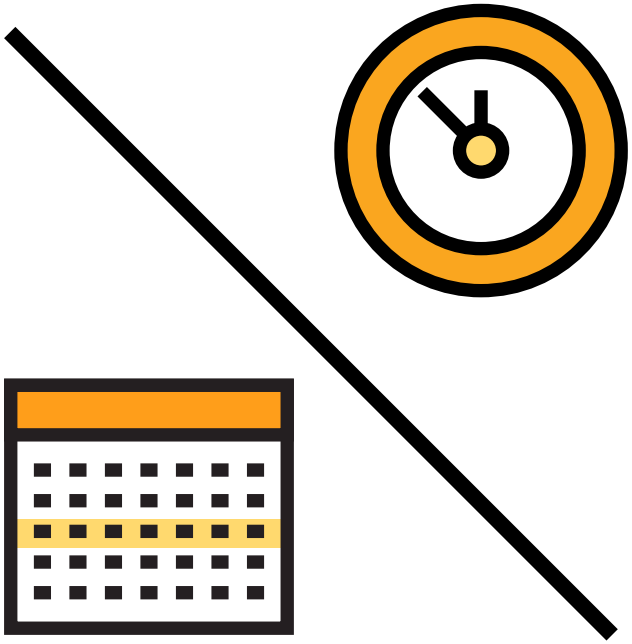
Trunk-based development

Last year, we investigated the role that trunk-based development plays in continuous delivery. While our experience shows that developers in high-performing teams work in small batches and **develop off of trunk or master**, rather than long-lived feature branches, many practitioners in the industry routinely work in branches or forks. Last year's results confirmed that the following development practices contribute to higher software delivery performance:

- Merging code into trunk on a daily basis.
- Having branches or forks with very short lifetimes (less than a day).
- Having fewer than three active branches.

We also found that teams without code lock periods had higher software delivery performance. (The ability to work without code lock periods is supported by the practices described above.)

Despite abundant evidence that trunk-based development practices contribute to better software delivery performance, some developers who are used to the **GitHub-recommended workflow** remain skeptical. The GitHub-recommended workflow suggests developing from branches, and only periodically merging to trunk. Working on short-lived branches that are merged into trunk at least daily is consistent with commonly accepted continuous integration practices. We have heard, for example, that branching strategies are effective, as long as development teams don't maintain branches for too long. But what is too long?



High performers typically experience branch life and integration lasting hours, versus days for low perfomers.

This year we conducted additional analysis to see if there is evidence for performance differences correlating to trunk-based development practices. We investigated how long branches and forks live before being integrated into master, and how much time it takes to merge and integrate the code on them. We wanted to see if there were significant differences between high, medium, and low IT performers. Here is what we found:

- High performers have the shortest integration times and branch lifetimes, with branch life and integration typically lasting hours.
- Low performers have the longest integration times and branch lifetimes, with branch life and integration typically lasting days.
- These differences are statistically significant.

This confirms last year's results, which showed that teams do better in terms of IT performance when they use short-lived branches in combination with short merging and integration periods.

There's clear guidance to offer from these findings: Teams should avoid keeping branches alive more than a day. If it's taking you more than a day to merge and integrate branches, that's a warning sign, and you should take a look at your practices and your architecture.

07

Lean product management

In last year's investigation of product management, we modeled lean product management as comprising two capabilities:

- Splitting work into small batches and making visible the flow of work through the delivery process.
- Gathering, broadcasting and implementing customer feedback.

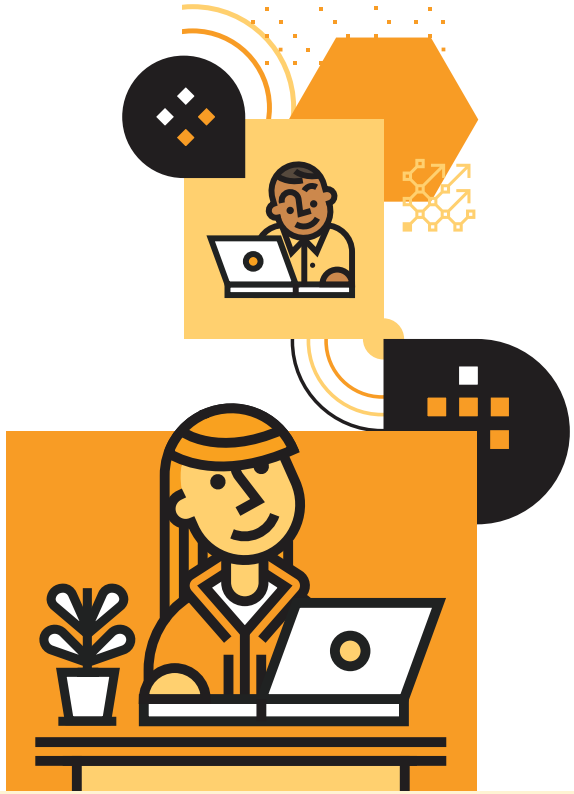
This year we extended our model to investigate the effect of a key agile principle: giving development teams the authority to create and change specifications as part of the development process, without requiring approval.

Working in small batches and gathering feedback

Last year, we learned that certain lean product management practices predicted higher IT performance and lower deployment pain. These practices include decomposing work into small batches; creating prototypes or minimum viable products (MVPs); and actively seeking customer feedback as an input for product design. We hypothesized that to become truly proficient in lean product management, you need to undergird it with a strong foundation in software delivery.

This year, we flipped the model and found that IT performance *predicts* lean product management practices. Improving your software delivery pipeline will improve your ability to work in small batches and incorporate customer feedback along the way. If we combine these models across years, it becomes a reciprocal model, or colloquially, a virtuous cycle.

In software organizations, the ability to work and deliver in small batches is especially important, because it allows you to gather user feedback quickly using techniques such as A/B testing. It's worth noting that the ability to take an experimental approach to product development is highly correlated with the technical practices that contribute to continuous delivery. You can learn more about these practices in the **Technical practices** section.



How to build a lean mindset

Are you thinking about MVPs the right way?

Read Henrik Kniberg's blog post:

blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp

Empowered development teams

Many development teams working in organizations that claim to be agile are nonetheless obliged to follow requirements created by different teams. This restriction can create some real problems, and result in products that don't actually delight and engage customers, and don’t deliver the expected business results.

One of the purposes of agile development is to seek input from customers throughout the development process, including the early stages. This allows the development team to gather important information, which then informs the next stages of development. If a development team isn't allowed to change requirements or specifications in response to what they discover, without authorization from some outside body, their ability to innovate is sharply curtailed.

Our analysis showed that a team's ability to try out new ideas and create and update specifications during the development process (without requiring approval from outside the team) is an important factor in predicting organizational performance, as measured in terms of profitability, productivity, and market share.

We’re not proposing that you just set your developers free to work on whatever ideas they like. To be effective, empowerment must be combined with the other capabilities we measure here: working in small batches; making the flow of work through the delivery process visible to everyone; and incorporating customer feedback into the the design of products. This ensures that your teams are making well reasoned, informed choices about the design, development and delivery work they do, and changing that work based on feedback. This increases the probability that the ideas and features they build will deliver delight to customers, and value to the organization.

DevOps & COTS

A common myth we hear is: “We can’t do DevOps in our environment — we’re mostly running commercial off-the-shelf software.” However, here's what we found in the **2015 State of DevOps Report**:

> *It doesn’t matter if your apps are greenfield, brownfield or legacy — as long as they are architected with testability and deployability in mind, high performance is achievable. We were surprised to find that the type of system — whether it was a system of engagement or a system of record, packaged or custom, legacy or greenfield — is not significant. Continuous delivery can be applied to any system, provided it is architected correctly.*

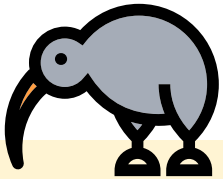
Of course, it's much easier to say you can apply DevOps practices and principles to commercial off-the-shelf software (COTS) than it is to actually do it. It's also true that many of the principles around how you develop software aren't going to be applicable to a COTS environment. But the benefits of applying some of the high-level deployment and testing practices will absolutely increase overall organizational velocity, improve service quality for your users, and increase productivity of your IT staff (while also helping them learn new skills, which is important both for retention and for your organization's ability to keep up with technology changes).

The critical first step is to understand the difference between *utility* and *strategic* projects, and to make sure that you approach them differently. **Martin Fowler distinguishes between utility and strategic projects** by whether or not the underlying function is a business differentiator. He argues that if there is no differentiation, it's better to install a package than to go build a custom solution. Many make the mistake of treating utility software as a custom solution, and expend effort on customizing it, rather than on adjusting business processes to the software.

If you approach your utility functions as if they're strategic, you're creating the worst of both worlds. You end up with customized black boxes that are expensive to maintain and test, painful to upgrade, and difficult to improve in an iterative manner. It's also near-impossible for you to automate the testing, deployment and maintenance stages of the service lifecycle — and automation is at the heart of DevOps practices.

Rather than customizing your off-the-shelf software to fit your perhaps ossified business processes, consider changing your business processes instead, to help promote optimal performance of the whole system. In many cases, it's far cheaper to modify your business processes, and use your off-the-shelf software as close to its out-of-the-box state as possible. In any case, it's always worth taking a look at your processes with fresh eyes at regular intervals.

It can go against the grain of engineering organizations to simply accept something other people built. Many engineers believe, "we can build it so much better ourselves." But if you can summon the humility to accept not-built-here, you can harness all the energy and talent in your organization to do what really matters: employing digital innovation to push your business forward and gain a real competitive advantage.



TelstraClear year-long upgrade

TelstraClear, a telecommunications company in New Zealand, had heavily customized its commercial service desk software, which made it difficult and time-consuming for the IT team to make system upgrades.

For more details, read Keep it simple, stupid: TelstraClear.

Conclusion

Over the past six years of surveying IT professionals and writing the DevOps Report, our team has made many breakthrough discoveries about the relationships between IT performance, technical practices, cultural norms and organizational performance. We've dispelled many DevOps myths, and been happily surprised to discover clear evidence in our data that DevOps practices yield remarkable results — even better than we expected. We've had some of our hypotheses disproved by the data, and we've learned from that, too.

In all our research, one thing has proved consistently true: Since nearly every company relies on software, IT performance is critical to any organization doing business today. And IT performance is affected by many different factors, including leadership, tools, automation and a culture of continuous learning and improvement.

We hope this report has helped you identify areas where you can improve your own IT and business processes, work culture, and learning cycles. We welcome your feedback: **devopssurvey@puppet.com**

Methodology

The State of DevOps Report has evolved over the past six years. Our rigorous methodology goes beyond reporting raw numbers and looks at the statistical relationships between IT performance, organizational performance, technical practices, cultural norms, and management. 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.

We welcome any questions about our survey methodology, so feel free to get in touch: devopssurvey@puppet.com.

How we determined high, medium, and low IT performance using cluster analysis

Rather than use predetermined levels to classify low, medium, and high performers, we use a data-driven approach. We do this because industry insights and benchmarks are key to real improvement. If we arbitrarily dictated performance levels for each metric, then after just a year or two, those who set performance targets by those levels could be lulled into a false sense of security while being outpaced by industry's truly innovative organizations, which know that the key to excellence is continuous improvement.

This is the fundamental flaw in maturity models: They outline static levels of performance that teams and organizations can target, but once you have "arrived," you risk stalling your improvement work and losing leadership support, because you have achieved your goal. By continually reassessing our IT performance groups based on current data, teams can understand what is happening in the industry.

So how do we determine our IT performance groups, using a data-driven approach? We use cluster analysis,[7] which gives us groupings of items — in this case, teams' metrics for deployment frequency, lead time for changes, MTTR, and change failure rate — that are statistically similar to those in the same group, and different from those in other groups. Using cluster analysis, we found distinct groupings and significant differences between companies.

Companies with high IT performance are similar to each other, and different from low- and medium-performing counterparts. Low IT performers are similar to other low IT performers, but different from high IT performers, and so on. For the fourth year in a row, this confirmed that high, medium, and low IT performers are statistically different from each other. (This is another weakness of maturity models: they are rarely or never indicative of real differences between groups.)

The benefit of cluster analysis is that it allows us to categorize groups without knowing ahead of time the values each group should have; it only requires the values should be significantly different. That is, we let data tell us how many groups there should be, and how to describe those groups using their average values. We have reported data to describe our high, medium, and low performers in Table 2, in the section titled IT performance & organizational performance.

7 "Cluster analysis," Wikipedia

Target population & sampling method

Our target population for this survey was practitioners and leaders working in, or closely with, IT, 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. Our sample is likely limited to organizations and teams that are familiar with DevOps, and as such, may be doing some of it.

Creating latent constructs

Once the target population and sampling method were determined, we began the difficult work of determining which questions to include in the survey. To do that, we first had to determine which hypotheses we wanted to test. To add to the rigor of our study, we referenced existing research and theories. We formulated our hypotheses and constructs, using previously validated constructs wherever possible. When we needed to create new constructs, we wrote them very carefully 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.[8] To create constructs, we used Likert-type[9] questions, which provided shades of gray, rather than black-and-white, yes-or-no, true-or-false questions. Likert-type questions also make it possible to perform more advanced analysis.

8 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.
9 "Likert scale," Wikipedia

Statistical analysis methods

- **Cluster analysis.** IT performance profiles are derived with a data-driven approach, using hierarchical cluster analysis. In this approach, those in one group are statistically similar to each other and dissimilar from those in other groups. For IT performance profiles, they are similar (or dissimilar) based on our IT performance behaviors of throughput and stability: deployment frequency; lead time; mean time to restore; and change fail rate.
- **Measurement model.** Prior to conducting any analysis using constructs — including correlations, regressions, and partial least squares[10] (PLS) analysis — the constructs were tested for validity and reliability. The constructs passed checks for convergent validity,[11] discriminant validity,[12] and reliability, therefore exhibiting good psychometric[13] properties.
- **Regression analysis.** When predictions or impacts are cited and PLS is not explicitly mentioned, a simple linear regression[14] was used.
- **Structured equation modeling.** The structured equation models (SEM)[15] were tested using PLS analysis, which is a correlation-based SEM well suited to exploratory analysis. SmartPLS 3.2.6 was used. All paths shown in Figure 1 are $p < .001$, except IT Performance → Lean and IT Performance → Organizational Performance, which are $p < 0.05$.
- **Study design.** This study employs a cross-sectional, theory-based design.

10 "Partial least squares regression," Wikipedia
11 "Convergent validity," Wikipedia
12 "Discriminant validity," Wikipedia
13 "Psychometrics," Wikipedia
14 "Linear regression," Wikipedia
15 "Structural equation modeling," Wikipedia

Acknowledgements

The authors would like to thank several external reviewers for their input and guidance on the items used to measure new constructs this year.

For the items measuring architecture, we thank **Neal Ford, Martin Fowler, Mik Kersten, Sam Newman,** and **Rebecca Parsons**.

For team experimentation, we thank **Amy Jo Kim** and **Mary Poppendieck**.

The authors would also like to thank **Aliza Earnshaw** for her meticulous work editing the State of DevOps Report for the past four years; the report would not be the same without her careful eye.

Author biographies

Dr. Nicole Forsgren is the CEO and chief scientist at DevOps Research and Assessment (DORA). She is an author and researcher in knowledge management, IT adoption and impacts, and DevOps, and is best known as the lead investigator on the largest DevOps studies to date. In a previous life, she was a professor, sysadmin, and hardware performance analyst. Nicole has been awarded public and private research grants (funders include NASA and the NSF), and her work has been featured in various media outlets and several peer-reviewed journals and conferences. She holds a Ph.D. in management information systems and a master's degree in accounting.

Jez Humble is co-author of 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, most recently working for the U.S. government at 18F. He is currently researching how to build high-performing teams at his startup, DevOps Research and Assessment LLC, and also teaches at UC Berkeley.

Gene Kim is a multiple-award-winning CTO, researcher and author, and has been studying high-performing technology organizations since 1999. Among the books he has co-authored are "The Phoenix Project" and "The DevOps Handbook." He is the primary organizer of the DevOps Enterprise Summit, and is a co-founder of DevOps Research and Assessment LLC.

Alanna Brown is director of product marketing at Puppet, where she's helped grow the marketing function and team for over five years. She conceived and launched the first annual State of DevOps Report in 2012, and has been responsible for the survey and report since then. In addition to heading up DevOps research, Alanna is also responsible for driving the go-to-market strategy for DevOps, cloud, and security partners; cultivating relationships with customers to drive DevOps adoption; and leading cross-functional teams to bring Puppet products to market.

Nigel Kersten is the chief technical strategist at Puppet, where he has held a variety of roles, including head of product, CTO and CIO. He came to Puppet from Google headquarters in Mountain View, Calif., where he was responsible for the design and implementation of one of the largest Puppet deployments in the world. Nigel has been deeply involved in Puppet's DevOps initiatives, and regularly speaks around the world about adoption of DevOps in the enterprise and IT organizational transformation.

About Puppet

Puppet is driving the movement to a world of unconstrained software change. Its revolutionary platform is the industry standard for automating the delivery and operation of the software that powers everything around us. More than 37,000 companies — including more than 75 percent of the Fortune 100 — use Puppet's open source and commercial solutions to adopt DevOps practices, achieve situational awareness and drive software change with confidence. Headquartered in Portland, Oregon, Puppet is a privately held company with more than 530 employees around the world.

Learn more at puppet.com.

About DevOps Research and Assessment

DevOps Research and Assessment (DORA), founded by Dr. Nicole Forsgren, Jez Humble and Gene Kim, conducts research into understanding high performance in the context of software development, and the factors that predict it. DORA's research over four years and more than 25,000 data points serves as the basis for a set of evidence-based tools for evaluating and benchmarking technology organizations, and identifying the key capabilities to accelerate their technology transformation journey.

Learn more at devops-research.com.