Nicole Forsgren, PhD

The Key to High Performance: *What the Data Says*

DevOps Enterprise Summit San Francisco 2017

The Key to High Performance

What the Data Says

DevOps Enterprise Summit San Francisco 2017

Nicole Forsgren, PhD

25 NW 23rd Pl, Suite 6314

Portland, OR 97210

The contents of this eBook are a transcript of the complete presentation given by Nicole Forsgren, "The Key to High Performance: What the Data Says" at the DevOps Enterprise Summit San Francisco 2017.
To view the original presentation, please visit https://www.youtube.com/watch?v=RBuPlMTXuFc&t=25s.

eBook published 2018 by IT Revolution Press.
For further information on this or any other books and materials produced by IT Revolution Press,
please visit our website at itrevolution.com

This eBook is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

The Key to High Performance: What the Data Says

Hasn't this been an amazing conference? Right? This is really one of my favorite places to be. As I was pulling together this talk, I was thinking back about the journey and where I've been, and the talk title I submitted was "The Key to High Performance and What the Data Says." But I was going through all of it and I realized it's really "How to Become a High Performer."

I was thinking back over the last few years. As Gene said, it's kind of been this four-year journey, and I thought back to the first DevOps Enterprise Summit. I stood on main stage, and I had this quick fifteen-minute almost TED Talk about what we found in that first State of DevOps Report. We talked about what it meant to be a high performer, and how we found that DevOps actually works, and how the ability to develop and deliver software with both speed and stability could really drive organizational outcomes.

By the way, this was kind of a big deal, and I'll get to that in just a minute. Okay, so then 2015 rolled around, and we really kind of expanded our study. We talked about how Lean management practices and principles drove excellence as well. In 2016, we branched out again. We really kind of shifted left again, talking about shifting left on security, test data management. 2017 we did a lot, but in 2017 we expanded again into what organizational performance really meant, because for some reason profitability, productivity, and market share just wasn't enough. We also wanted to know about effectiveness and efficiency and customer satisfaction.

As Gene mentioned, this research has been done with the core team at Puppet, me, and Gene [Kim], and Jez Humble, also the group at Puppet, as he mentioned. So, this journey has also taken us other places. We have some white papers talking about forecasting the value of DevOps transformations, which some of us call ROI. Who here cares about ROI? There's an executive who cares about ROI. That's for free. So, you've gone on our website, and it talks about how to access this. I've got a page at the end that lists a whole bunch of websites and stuff. We've worked with teams like Capital One. We have this book,[1] which if you want to really dig into it, we had a signing last night, but this really digs into the detail of the four years. But, for people who don't want to dig into four years and want to watch a video instead, I was thinking this might be a great opportunity for me to race through some of the highlights, but I'll warn you this can't cover everything.

These are some of my favorites over the last four years. Here's a quick outline:

- Technology matters.
- Maturity models don't work—It's fine trust me, I'll get to that one.
- Transformations need technology AND process AND culture.
- Also, architecture and technology.
- You can accelerate your journey.
- Leadership matters.
- And then, how you can help.

Let's get started. First of all, technology matters, which I'm sure everyone here is like, "Yeah, okay Nicole, seriously we know." But guess what, it didn't always matter, or at least not in the way we thought it did, or we want it to matter. In 2003, Nicholas Carr wrote this paper called "IT Doesn't Matter,"[2] and the challenge is that he wrote it in *Harvard Business Review*. All of our executives read this paper and then believed it and then decided that IT was a cost center. Who here has heard this? That sucks, right? That sucked really bad, but it was sort of true, and I'll get back to that in just a second.

It was actually based on the last ten or twenty…the previous ten or twenty or thirty or forty years of research. Suddenly, we had done this research in 2014, that was different. It was finding different things, and here's why. We were finding that DevOps is good for technology. We had to start with a definition, okay. DevOps is good for technology and suffer delivery performance. What do I mean by that? I mean the ability to develop and deliver software with both speed and stability.

We're in DevOps right, so develop and deliver software with both speed and stability. This is how we measure it: speed is deployment frequency, or when the business demands. That's important right, because now it's a business decision and not a technical constraint. Also, lead time for changes. Code commit to code deploy. This is the most predictable part of your pipeline. Now, stability, MTTR and change-fail rate. Now, here's the really cool part. Definitely pay attention to this. We find that software delivery performance is comprised of throughput and stability and both are possible without trade-offs. Four years of data, four years of analysis, over twenty-three thousand data points, they all move together. High performers, I do this analysis, this is called cluster analysis, they [high performers] all cluster together, and then I see a gap. Then the medium performers all cluster together, and then I see a gap. Then there's a group that all kind of sucks. It's fine, there's three groups and I'm super creative, so I call them high, medium, and low performers. But, they all group together. There aren't trade-offs. I don't see it. Anyone who's been telling you that in order to go fast you have to sacrifice stability, or in order to be stable you have to sacrifice speed, it's not true. I don't see it anywhere in the data.

Also, DevOps is good for organizations. Here's what we find. High performers year over year are twice as likely to achieve or exceed their commercial goals. Again, profitability, productivity, and market share. I like money. Any organization…anyone here who works for an organization that likes money? Profit driven? Right? This speaks to us. But it's interesting, after a few years some people came back to us and they said, "I work for a not for profit. I work for government. I work for some educational institutions," or "I work for a for profit institution that has broader goals. We also care about other things. Other things matter to us. What about non-commercial goals, even if I'm in the for-profit sector?"

In the 2017 study we added additional measures, like non-commercial goals and the crazy thing is we find the same 2x multiplier. High performing organizations are twice as likely to achieve or exceed non-commercial goals: Effectiveness, efficiency, customer satisfaction. By the way, the measures for these are drawn from the academic literature, highly rigorous measures.

For bonus points, in the 2014 study we had a large enough data set that I could actually pull from stock market data. We find that high performers had a fifty percent higher market cap growth over the previous three years.

This is great. Okay? The ability to develop and deliver software… technology is driving organizational performance.

The next one for me, maturity models don't work. They're not a great fit for us. Who here has played World of Warcraft? Okay, who has a friend that has played World of Warcraft? It's fine. When World of Warcraft first came out, early 2000's, this was the world. This [Figure 1] was the land. It went to level sixty, right? This was about as good as you could get.

Figure 1

Let's create a maturity model for World of Warcraft. This is sweet. Let's say level forty is good. Level fifty is rad. Level sixty's amazing. I'm making this up. Let's roll with it, it's great. Fast forward a few years and the landscape has changed. [See Figure 2] We have extra worlds. We have extra lands. You can now also get to level one hundred ten. There are extra tooling and technologies available. Does this sound like technology?

Figure 2

If we create maturity models, they go out of date too fast. This is a challenge. The technology landscape has changed in the last ten years. Is this sounding familiar? There's new tooling. There's new technology. The competitive landscape has changed. If we rely on a maturity model that was created five years ago, ten years ago, it's out of date. Also, if we reach that magical level four or magical level five, what happens when resources disappear? Those resources can be money. They can be time. What if they're attention? What happens when our executives or leadership lose attention to our transformation?

The same is true in the data I've collected over only the last four years. Let's not even go back a decade, let's go back four years. The year-over-year data trends are drastically different. What is good enough in one year is completely out of date the next. If I had created a maturity model of software development delivery in 2014, it would've been wrong and out of date in 2015. It would have been old in 2016 and it would've been old in 2017. I'm not doing it in 2018 either.

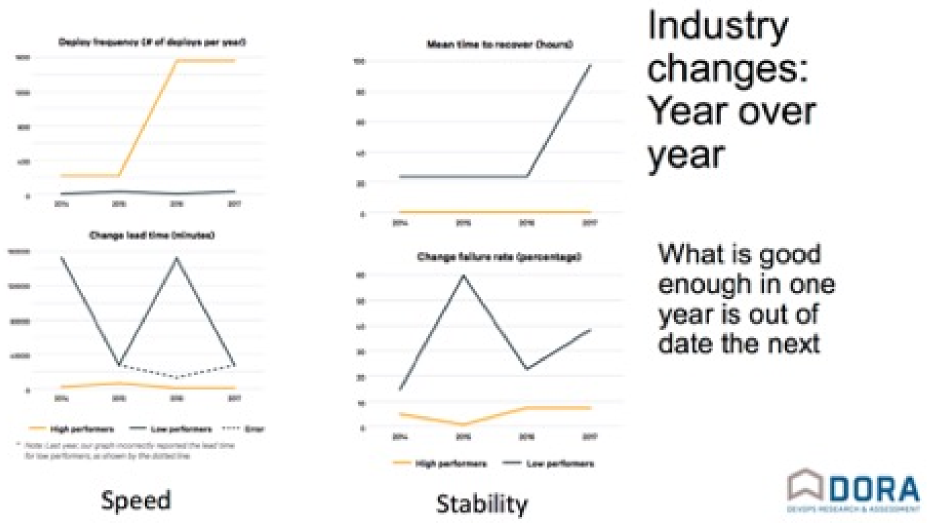
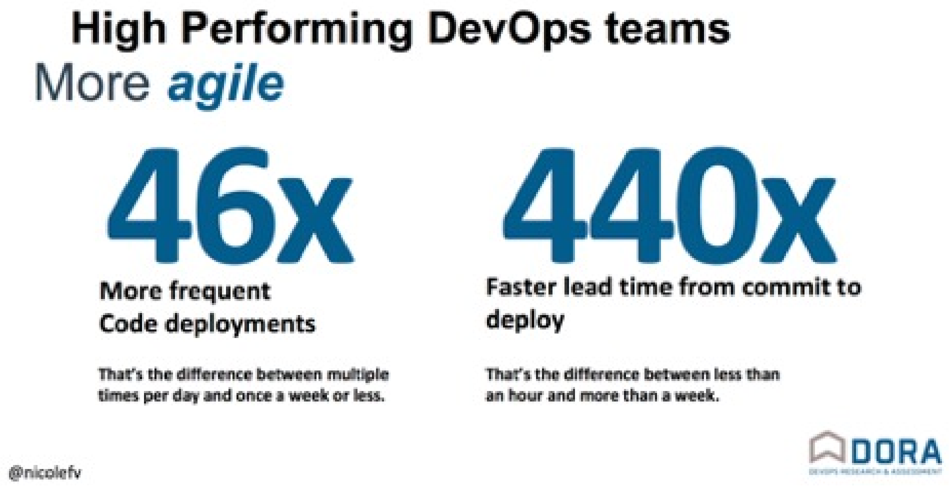


Figure 3: Industry Changes: Year over Year

Let's look at 2017's data. High performing DevOps teams are more agile. We see forty-six times more frequent code deployments. This is the difference between deploying code multiple times a day and once a week or less. And by the way, this is just typical performance. We still have people that are doing this much less. We also see four hundred forty times faster lead time from code commit to code deploy. It's the difference between less than an hour and more than a week. How fast can you get those changes through your pipeline? This is how we can delight our customers, beat our competitors to market, or on the off chance you have no competitors, how you can keep up with compliance and security changes. This is the power that this gives us.



High Performing DevOps teams

More *agile*

46x

More frequent Code deployments

That's the difference between multiple times per day and once a week or less.

440x

Faster lead time from commit to deploy

That's the difference between less than an hour and more than a week.

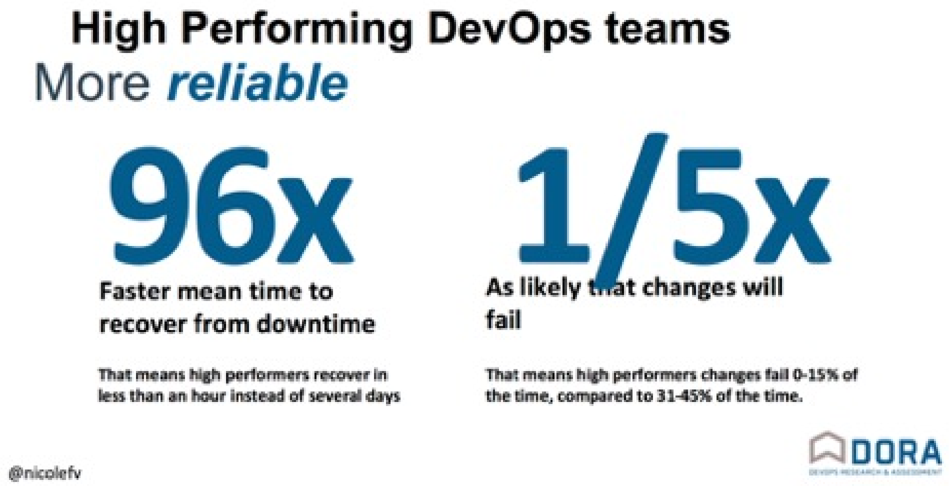
@nicolefv

DORA

Figure 4

Let's move over to stability. Our high performing DevOps teams are also more reliable. We see ninety-six times faster mean time to recover from down time. High performers are recovering in less than an hour instead of several days. Y'all, days. Don't raise your hand, who is down for days? Don't raise your hand. Slow is the new down. Being down is tough. We also see that high performing teams are one-fifth as likely that their changes will fail. When they do fail, they have a much slower down time, they have a much smaller blast radius, and it's much easier to identify what those changes are because the changes that went through that pipeline are so much smaller. It's very easy to identify what went wrong.

This is my favorite eye chart. Luckily, we have very big screens. Here [Figure 6] we can see what the high performers are doing, what the medium performers are doing, and what the low performers are doing in 2017. The thing to take away here is to notice that the high performers are really maximizing across all dimensions. Remember how I said there are no trade-offs? I don't see trade-offs. Take a look. See where your teams are.



High Performing DevOps teams

More *reliable*

96x

Faster mean time to recover from downtime

That means high performers recover in less than an hour instead of several days

1/5x

As likely that changes will fail

That means high performers changes fail 0-15% of the time, compared to 31-45% of the time.

@nicolefv

DORA

Figure 5

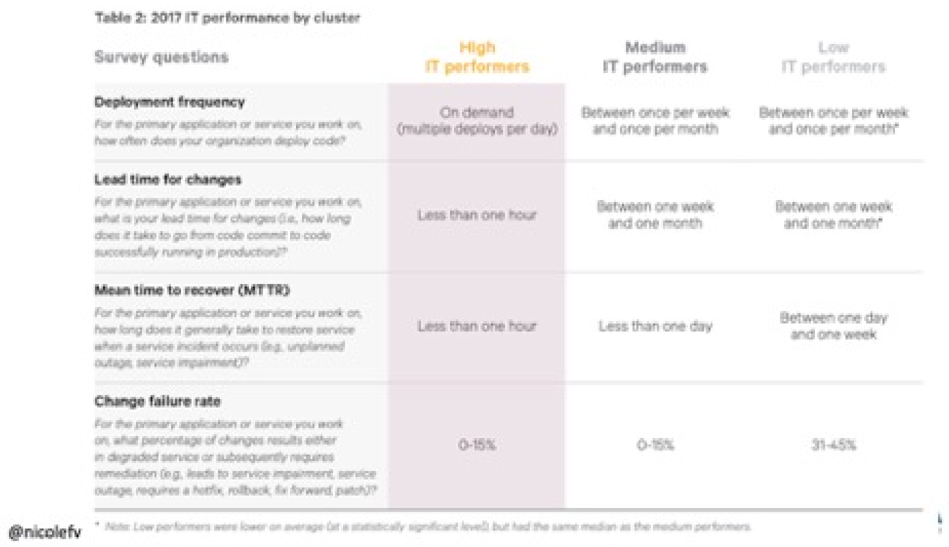


Table 2: 2017 IT performance by cluster

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%

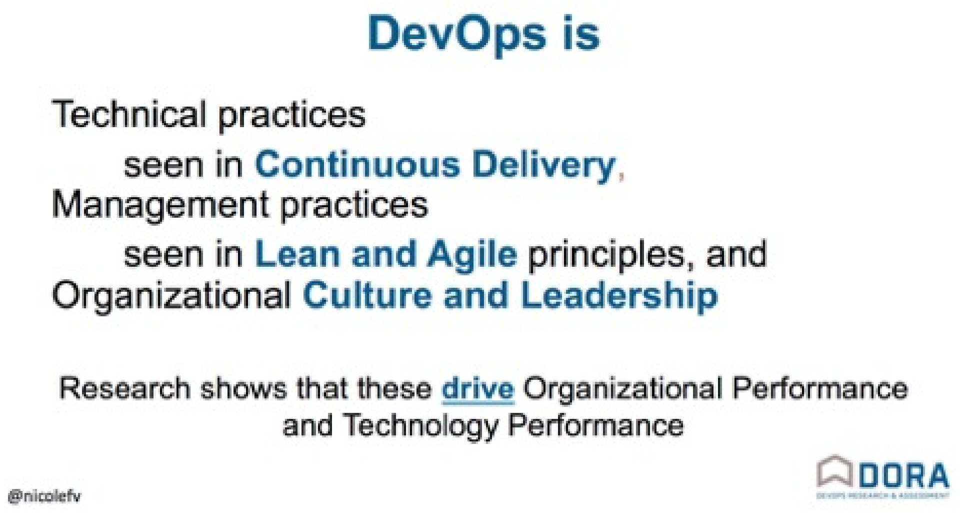
* *Note: Low performers were lower on average (at a statistically significant level) but had the same median as the medium performers.*

@nicolefv

Figure 6

By the way, I do categorize this at the team level because, particularly in large organizations, different teams will perform at different levels, and we expect that. We've seen that and we've heard that here across so many of our talks. Also, I'm going to mention, this data that we've collected over all these years covers organizations across all industries, all sizes, around the world. Subtext, there's no excuse. I see all industries in high performers. I also see all industries in low performers. You can do this, and DevOps is there to help.

Next up, transformations need technology and process and culture. How we've categorized it is technical process seen in continuous delivery, management practices and capabilities seen in Lean and Agile principles. We've drawn from these movements and organizational culture and leadership. Again, we've seen that these drive organizational performance and technology performance.



DevOps is

Technical practices
seen in Continuous Delivery,
Management practices
seen in Lean and Agile principles, and
Organizational Culture and Leadership

Research shows that these drive Organizational Performance and Technology Performance

@nicolefv

DORA

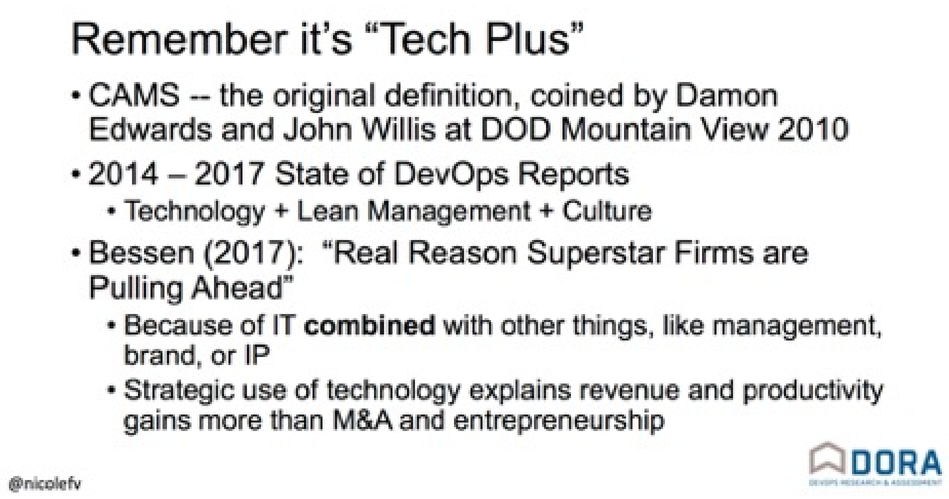
Figure 7

Remember Nicholas Carr? The guy that said that IT doesn't matter? He sort of had a point, but here's why. Back then, back in the 80's and 90's, his data really drew from the 80's and 90's, we used to buy technology, throw it in a closet, check a box, and walk away. Does anyone remember that? None of us here are old enough for that, but maybe I remember that, totally. It's fine, I remember that.

The challenge is that when we do technology alone for technology sake, it's not really impactful. It's not super strategic, right? If we only do the technology and then walk away it doesn't have these big impacts. We only do technology. We need technology *and* process *and* culture to really get this strong synergy. Then we only have a point of parody we can keep up. Of course we should be using technology because we shouldn't be doing things by hand. Don't do accounting by hand, don't be doing math by hand, don't be doing HR systems by hand. It gives us a point of parody, but everyone else is buying computer systems. It doesn't get us a point of distinction. It doesn't get us ahead.

Who here has maybe heard of buying DevOps in a box or the mistaken belief that DevOps is buying a new technology or tooling system? That's getting us back to the 80's and 90's again. That's why it's not going to work. DevOps is a challenge. DevOps is hard, but that's also why suddenly we're seeing it deliver real value to organizations. Always remember, it's tech plus.

Has anyone here heard CAMS? This original definition really kind of coined by Damon Edwards and John Willis at DevOps Days Mountain View 2010. I'm also a fan of CALMS, a culture automation lean measurement and sharing, I like that 'l' in there. The 2014 to 2017 State of DevOps Reports found this as well, that tech plus drives performance gains. Technology *plus* Lean management *plus* culture.



Remember it's "Tech Plus"

- CAMS -- the original definition, coined by Damon Edwards and John Willis at DOD Mountain View 2010
- 2014 – 2017 State of DevOps Reports
  - Technology + Lean Management + Culture
- Bessen (2017): "Real Reason Superstar Firms are Pulling Ahead"
  - Because of IT **combined** with other things, like management, brand, or IP
  - Strategic use of technology explains revenue and productivity gains more than M&A and entrepreneurship

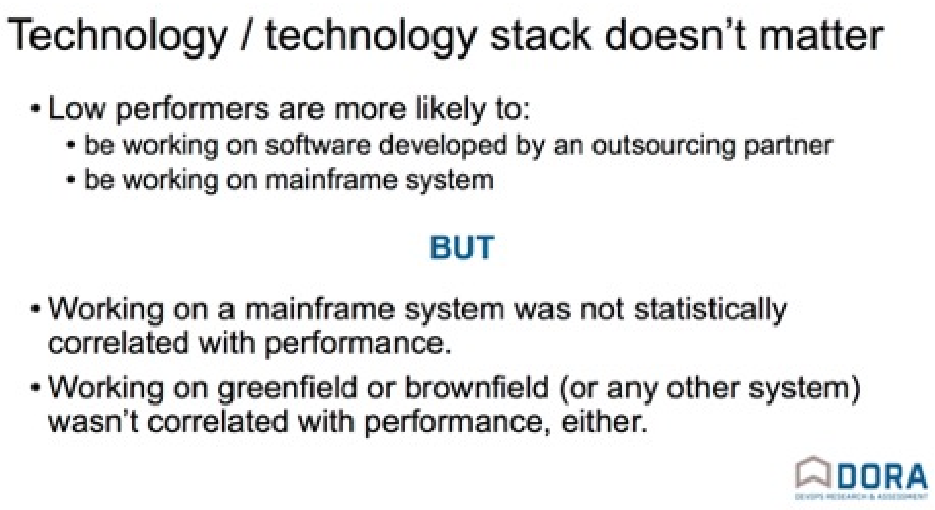
@nicolefv

DORA

Figure 8

If you don't want to trust me, look at a very recent article by Bessen. He comes from Boston University. I love this, a summary, a really accessible summary, it was just published in HBR, *Harvard Business Review*. The title is "The Real Reason Superstar Firms are Pulling Ahead," and it's because of IT combined with other things like management, brand, or IP. It said "The strategic use of technology combined explains revenue and productivity gains more than MNA and entrepreneurship."[3] Again, it's tech plus.

Next up, architecture matters, technology doesn't. We find that low performers are more likely to be working on software developed by an outsourcing partner. They're more likely to be working on mainframe systems. We found this in the 2015 study. But guess what? It wasn't statistically significant. There was no significant correlation with performance from working on a mainframe system. There was also no statistically significant correlation for working on greenfield and brownfield systems either. Do you know what was correlated? Having a good architecture. If you have well-architected systems, it doesn't matter what type of platform you're working on.



Technology / technology stack doesn't matter

- Low performers are more likely to:
  - be working on software developed by an outsourcing partner
  - be working on mainframe system

BUT

- Working on a mainframe system was not statistically correlated with performance.
- Working on greenfield or brownfield (or any other system) wasn't correlated with performance, either.

DORA

Figure 9

In 2017 we dug into this. Here's what we find. If you can answer these questions positively this is what counts.

- Can your team make large scale changes to the design of your systems without the permission of someone outside of your team or without depending on other teams?
- Can your team complete its work without fine-communication and without fine-grained coordination?
- Can your team deploy and release your product on demand independently of other services?
- Can your team do most of your testing on demand without requiring an integrated test environment?

- Can your team perform deployments during normal business hours and with negligible down time?

The thing that is most important is having a loosely coupled architecture both at the team level and at the technology level. It's interesting. Sounds a lot like Conway's Law. Organizations which design systems are constrained to produce designs which are copies of the communication structures of those organizations. Having smartly designed API's, having good service-oriented architectures, are good investments in organizations. It doesn't matter what type of architecture you have. We've seen great organizations have mainframes that are doing fantastic work.

Next up, you can accelerate your journey. You can accelerate your transformation. These are the key capabilities that really drive software delivery performance. They fall into four categories:

1. tech and automation
2. process
3. measurement and monitoring
4. and culture

Quick take a picture. These are the ones that we have seen are predictive of your ability to develop and deliver software with both speed and stability. This is really kind of the core of our research. Technology and automation:

- version control for all production artifacts,
- deployment automation,

- continuous integration,
- trunk-based development,
- test automation,
- test data management,
- shifting left on security,
- continuous delivery,
- having a loosely coupled architecture that I just talked about,
- and then, architecting for empowered teams.

Process:

- gathering and implementing customer feedback,
- working in small batches,
- having a light weight change approval process,
- and allowing teams to experiment.

One example here, we did some work with Capital One, and they found that by focusing on two core things, for them it was trunk-based development and streamlining that change approval process, they saw stunning improvements. In just two months they saw a 20x increase in the number of releases with zero increase in incident.

Moving on to measurement and monitoring:

- visual management,
- monitoring for business decisions,
- checking system health proactively,
- use of WIP limits
- and visualizations.

These are all predictive, again, of that ability to develop and deliver software with both speed and stability.

Back to Capital One, they have Hygeia, which is an open source solution to help communicate visualizations throughout teams across their organization.

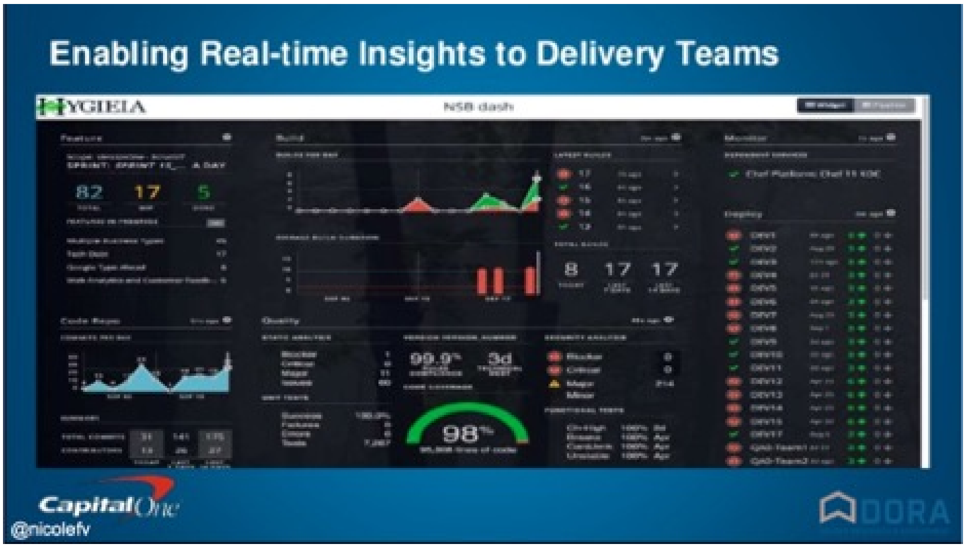
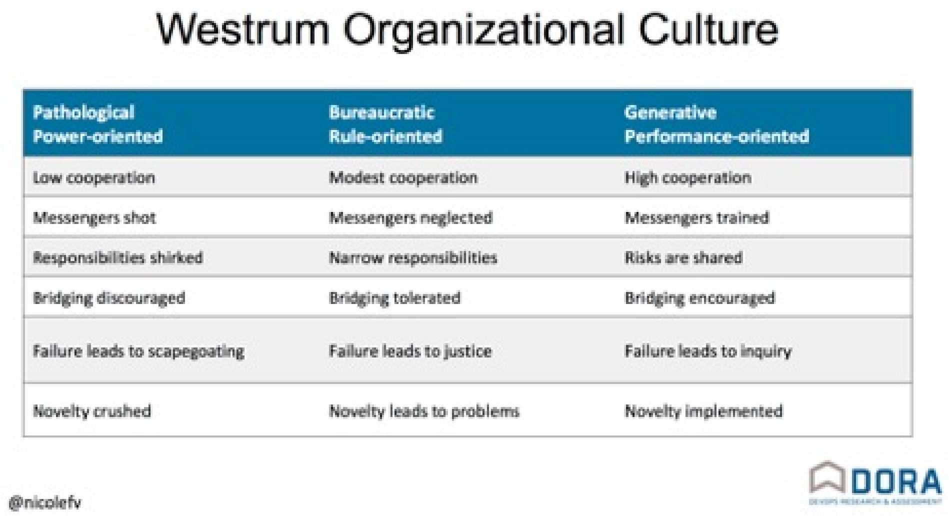


Figure 10: Hygeia Dashboard

Okay, culture. I'm sure we've all heard that tech is easy and people are hard, and cultures super important.

- Westrum organizational culture, which I'll get back to,
- having a climate for learning, so learning is seen as an investment not a cost,
- having good collaboration among teams,
- making work that's meaningful,
- and transformational leadership.

Westrum organizational culture, in particular, has been a really strong indication and prediction for performance. The reason that we chose this is because it had such strong predictive ability among highly complex work environments. If anyone has ever seen me talk you've probably heard me talk about this.



Westrum Organizational Culture

Pathological Power-oriented	Bureaucratic Rule-oriented	Generative Performance-oriented
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

@nicolefv

DORA

Figure 11: Westrum Organizational Culture

Ron Westrum is actually a sociologist, he's a PhD and he studies high-risk, complex work environments, like healthcare, aviation, and we see that this really kind of maps to what we mean when we talk about a DevOps culture. We tend to talk about doing blameless post-mortems. Failure leads to inquiry. We talk about breaking down silos and talking to people in other groups so bridging's encouraged. We talk about tackling problems and working out novel solutions and innovating so novelty's implemented. When we implement this and when we tested this, we found that this was highly correlated with each of our performance profiles. The pathological group ends up being the most highly correlated with low performance, bureaucratic is most highly correlated with medium performance, and the generative group is most highly correlated with high performance.

We also see that this is highly predictive of both software delivery performance and organizational performance. Investing in creating a good, high-trust culture is a great way to impact your organization. The question I often get here is, "Where do I start?" The right answer, I'm sure we all know is, "It depends." Everyone's different. But based off the data I see, here are a couple hints or cliff notes.

Based on the 2017 data, architecture was the highest contributor to continuous delivery. I actually kind of dedicated that whole section to having a loosely coupled architecture. Based on the DORA assessments that we do, we also see that loosely coupled architecture and a trunk-based development ends up being a constraint for so many teams that we work with. That's probably going to be a great place to start looking.

The probably number one or number two biggest constraint for most teams is a lightweight change approval process. Always look there. Do you have a lot of handoffs? Do you end up losing a lot of time there? Check it out. The last one is probably continuous integration and that full compliment. Also, pro tip, go dig into continuous integration and what it actually means, not what you have maybe defined continuous integration to mean. Turns out, almost everyone has their own definition for CI.

So, what do you do? Here to improve? Identify your constraints. Figure out what's slowing you down. What's your bottleneck? Pick a few. Be aware of death by initiative. There are a handful of companies that I talk to that have fifty initiatives, whoa. Pick a few. Work to eliminate those constraints and then re-evaluate. Once you've eliminated four or five, number six is probably no longer number one. We work in complex systems, once we've eliminated a few, everything will reshuffle. Then rinse and repeat.

Next up, leadership matters. We dug into transformational leadership in 2017, and we selected this model because this is the one that is the strongest, most strongly predictive. This is born out of a lot of the research, it's a five dimensional model. It's based on intellectual stimulation, inspirational communication, supportive leadership, personal recognition, and vision. What we find is that transformational leaders are really able to drive, support, and amplify the work that happens in organizations.

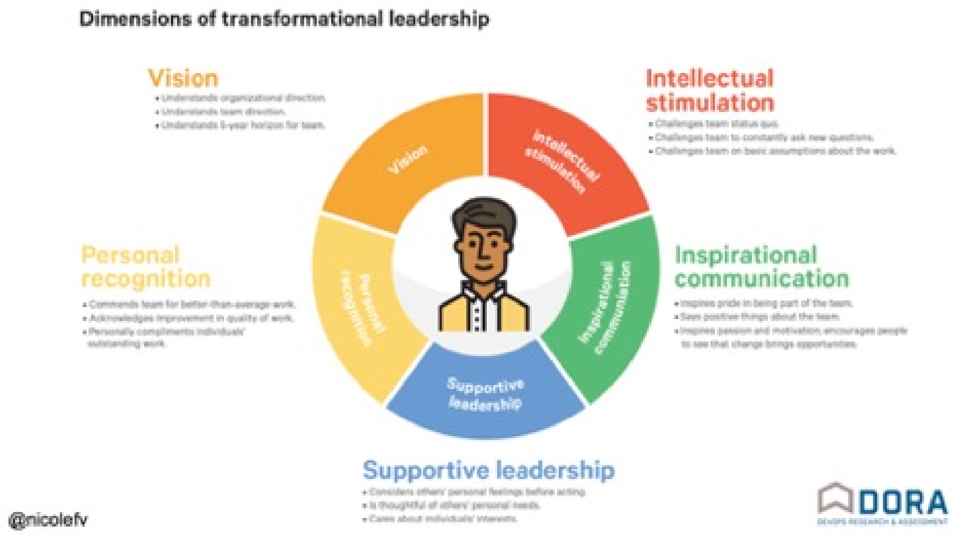

Dimensions of transformational leadership
Vision
• Understands organizational direction.
• Understands team direction.
• Understands 5-year horizon for team.
Intellectual stimulation
• Challenges team status quo.
• Challenges team to constantly ask new questions.
• Challenges team on basic assumptions 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.
Vision
Intellectual stimulation
Inspirational communication
Supportive leadership
Personal recognition
@nicolefv
DORA
DEVOPS RESEARCH & ASSESSMENT

Figure 12: Dimensions of Transformational Leadership

We also see that leadership is necessary but not sufficient. Teams with the least transformational leaders, that bottom third, were only one half as likely to be a high performer. But leaders can't do it alone. Teams at the very top performed no better than the median. What does that actually look like?

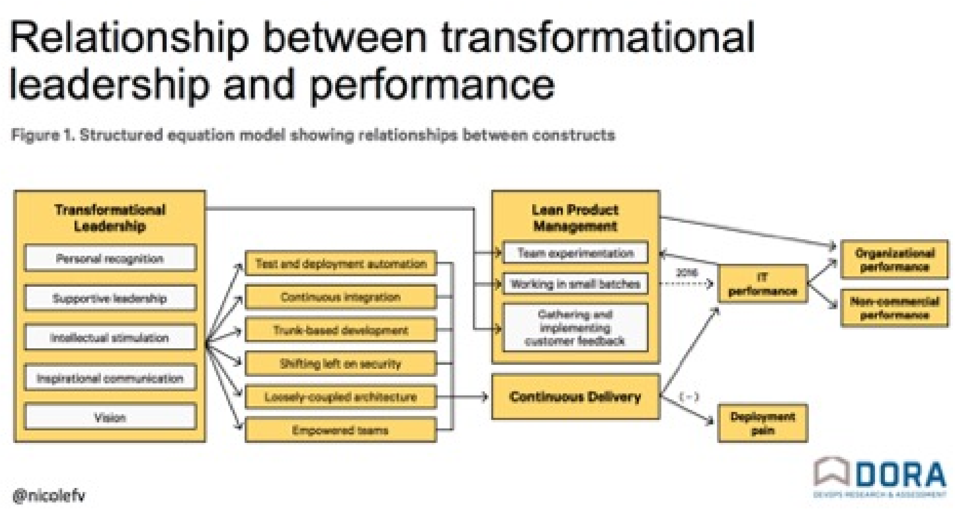

Relationship between transformational leadership and performance
Figure 1. Structured equation model showing relationships between constructs
Transformational Leadership
Personal recognition
Supportive leadership
Intellectual stimulation
Inspirational communication
Vision
Test and deployment automation
Continuous integration
Trunk-based development
Shifting left on security
Loosely-coupled architecture
Empowered teams
Lean Product Management
Team experimentation
Working in small batches
Gathering and implementing customer feedback
Continuous Delivery
2016
IT performance
(-)
Deployment pain
Organizational performance
Non-commercial performance
@nicolefv
DORA
DEVOPS RESEARCH & ASSESSMENT

Figure 13

This [Figure 13] was the research model for 2017. I won't walk you through the whole thing, but any time you see an arrow you can read that as a prediction relationship. We go well beyond correlation into prediction but look at where transformational leadership lies. That doesn't mean it's the most important thing, but it does mean that it supports and it underlies and it sort of amplifies everything else that happens.

Look at how important we are all in here. We have the ability to drive and support all of that technical work in testing deployment automation, continuous integration, trunk-based development, shifting left on security, that whole technical stack. We also have the ability to drive and support Lean product management. And then look at all of the arrows that come after that work. After that comes IT performance, a decrease in deployment pain, and then organizational performance. We support and amplify all of that work. We're at the core.

Good practices and smart investment make the work better too. The work, what do I mean by the work? Less deployment pain, less burn out, higher employee net promoter score. Who here's hiring? Who's organization is hiring? Okay, every hand right? We're all hiring. Employees in high performing organizations are 2.2 times more likely to recommend their organization as a great place to work. Other research coming out of HBR shows that high ENPS scores, that employee net promoter score, drives revenue as well. The investments are worth it.

You can help. Be the transformational owners, or sorry, transformational leaders. Own this. We can really make a big difference. Start your own measurement journey, which now someone might say, "Wait, measurement? She hasn't been talking about measurement." We don't know where we are unless we measure a few things. We can track where we are on our journey.

Start with five things. Make sure you focus a little bit on outcomes. Maybe software delivery performance. That ability to develop and deliver software with speed and stability. Those were outlined earlier, the slides will be up. Then also, thinking about driving performance improvements. Tech for sure, but also think about things that are not tech. Something that's process, something that's culture, and then share your stories. Share with each other for sure. This community is super important. Then reach out, share with me. Let me know what's working. Let me know what's not working. Let me know what your headaches are. It's time to design the 2018 State of DevOps Report and State of DevOps survey. I'm dying to know what's happening. I always love to know what's happening.

For more information, you can always reach out to our website, devops-research.com. We have white papers, we have case studies, we have links to all of the research. I just went through some of my favorites because Gene only gave me thirty minutes. There's so much there. Thank you so much for all of your time.

Endnotes

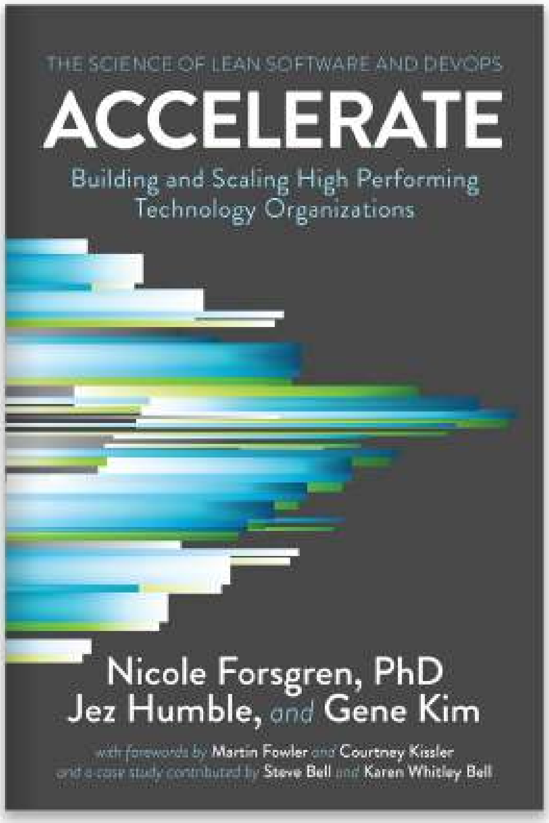
1. Nicole Forsgren, PhD, Jez Humble, and Gene Kim, *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations* (Portland, OR: IT Revolution Press, 2018).
2. Nicholas G. Carr, "IT Doesn't Matter," *Harvard Business Review*, May 2003, https://hbr.org/2003/05/it-doesnt-matter.
3. Walter Frick, "The Real Reason Superstar Firms are Pulling Ahead," *Harvard Business Review*, October 2007, https://hbr.org/2017/10/the-real-reason-superstar-firms-are-pulling-ahead.

About the Author

Dr. Nicole Forsgren is an IT impacts expert who is best known for her work with tech professionals and as the lead investigator on the largest DevOps studies to date. She is a consultant, expert, and researcher in knowledge management, IT adoption and impacts, and DevOps. In a previous life, she was a professor, sysadmin, and hardware performance analyst. She 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 PhD in management information systems and a masters in accounting.

Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations

By Nicole Forsgren, PhD, Jez Humble, and Gene Kim


THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell

Now available in paper, eBook, and audio formats from all major retailers: https://itrevolution.com/book/accelerate/.