Praise for *Accelerate*

"This is the kind of foresight that CEOs, CFOs, and CIOs desperately need if their company is going to survive in this new software-centric world.

Anyone that doesn't read this book will be replaced by someone who has."

—Thomas A. Limoncelli, coauthor of *The Practice of Cloud System Administration*

"'Here, do this!' The evidence presented in Accelerate *is a triumph of research, tenacity, and insight, proving not just correlation but a causal link between good technical and management behaviors and business performance. It also exposes the myth of 'maturity models' and offers a realistic, actionable alternative. As an independent consultant working at the intersection of people, technology, process, and organization design this is manna from heaven!*

As chapter 3 concludes: 'You can act your way *to a better culture by implementing these practices in technology organizations' [emphasis mine]. There is no mystical culture magic, just 24 concrete, specific capabilities that will lead not only to better business results, but more importantly to happier, healthier, more motivated people and an organization people want to work at. I will be giving copies of this book to all my clients."*

—Dan North, independent technology and organization consultant

"Whether they recognize it or not, most organizations today are in the business of software development in one way, shape, or form. And most are being dragged down by slow lead times, buggy output, and complicated features that add expense and frustrate users. It doesn't need to be this way. Forsgren, Humble, and Kim shine a compelling light on the what, why, and how of DevOps so you, too, can experience what outstanding looks and feels like."

—Karen Martin, author of *Clarity First* and *The Outstanding Organization*

"Accelerate *does a fantastic job of explaining not only what changes organizations should make to improve their software delivery performance, but also the why, enabling people at all levels to truly understand how to level up their organizations."*

—Ryn Daniels, Infrastructure Operations Engineer at Travis CI and author of *Effective DevOps*

"The 'art' of constructing a building is a well-understood engineering practice nowadays. However, in the software world, we have been looking for patterns and practices that can deliver the same predictable and reliable results whilst minimising waste and producing the increasingly high performance our businesses demand.

Accelerate *provides research-backed, quantifiable, and real-world principles to create world-class, high-performing IT teams enabling amazing business outcomes.*

Backed by the two leading thought leaders (Kim and Humble) in the DevOps community and world-class research from PhD Forsgren, this book is a highly recommended asset!"

—Jonathan Fletcher, Group CTO, Hiscox

"In their book Accelerate*, Nicole Forsgren, Jez Humble, and Gene Kim don't break any new conceptual ground regarding Agile, Lean, and DevOps. Instead, they provide something that might be even more valuable, which is a look inside the methodological rigor of their data collection and analysis approach which led them to their earlier conclusions on the key capabilities that make IT organizations better contributors to the business. This is a book that I will gladly be placing on my bookshelf next to the other great works by the authors."*

—Cameron Haight, VP and CTO, Americas, VMware

"The organizations that thrive in the future will be those that leverage digital technologies to improve their offerings and operations. Accelerate *summarizes the best metrics, practices, and principles to use for improving software delivery and digital product performance, based on years of well-documented research. We strongly recommend this book to anyone involved in a digital transformation for solid guidance about what works, what doesn't work, and what doesn't matter."*

—Tom Poppendieck and Mary Poppendieck, authors of the *Lean Software Development* series of books

"With this work, the authors have made a significant contribution to the understanding and application of DevOps. They show that when properly understood, DevOps is more than just a fad or a new name for an old concept. Their work illustrates how DevOps can improve the state of the art in organizational design, software development culture, and systems architecture. And beyond merely showing, they advance the DevOps community's qualitative findings with research-based insights that I have heard from no other source."

—Baron Schwartz, Founder and CEO of VividCortex and coauthor of *High Performance MySQL*

ACCELERATE

THE SCIENCE BEHIND DEVOPS

ACCELERATE

Building and Scaling High Performing Technology Organizations

Nicole Forsgren, PhD
Jez Humble *and* Gene Kim

IT Revolution
Portland, Oregon

IT REVOLUTION
25 NW 23rd Pl, Suite 6314
Portland, OR 97210

First Edition
Printed in the United States of America
22 21 20 19 18   1 2 3 4 5 6

Cover and book design by Devon Smith Creative, LLC

Library of Congress Catalog-in-Publication Data is available upon request.
ISBN: 978-1942788331
eBook ISBN: 978-194278355
Kindle ISBN: 978-194278362
Web PDF ISBN: 978-194278379

ACCELERATE

Contents

Figures

Tables

FOREWORD

By Martin Fowler

A few years ago I read a report that said, "We can now assert with confidence that high IT performance correlates with strong business performance, helping to boost productivity, profitability, and market share." When I read something like that, my usual response is to toss it with great force into the rubbish bin, because that's usually a tell for some bogus bullshit masquerading as science. I hesitated this time, however, for this was the "2014 State of DevOps Report." One of its authors was Jez Humble, a colleague and friend who I knew was equally allergic to this kind of twaddle. (Although I have to confess that another reason for not tossing it was that I was reading it on my iPad.)

So, instead I emailed Jez to find out what lay behind this statement. A few weeks later I was on a call with him and Nicole Forsgren, who patiently walked me though the reasoning. While I'm no expert on the methods they used, she said enough to convince me there was some real analysis going on here, far more than I usually see, even in academic papers. I followed the subsequent State of DevOps reports with interest, but also with growing frustration. The reports gave the results of their work but never contained the explanation that Nicole walked through with me on the phone. This greatly undermined their credibility, as there was little evidence that these reports were based on more than speculation. Finally, those of us that had seen behind the curtains convinced Nicole, Jez, and Gene to reveal their methods by writing this book. For me, it's been a long wait, but I'm glad I now have something that I can genuinely recommend as a way to look at IT delivery effectiveness—one that's based on more than a few analysts' scattered experiences.

The picture they paint is compelling. They describe how effective IT delivery organizations take about an hour to get code from "committed to mainline" to "running in production," a journey lesser organizations take months to do. They, thus, update their software many times a day instead of once every few months, increasing their ability to use software to explore the market, respond to events, and release features faster than their competition. This huge increase in responsiveness does not come at a cost in stability, since these organizations find their updates cause failures at a fraction of the rate of their less-performing peers, and these failures are usually fixed within the hour. Their evidence refutes the bimodal IT notion that you have to choose between speed and stability—instead, speed depends on stability, so good IT practices give you both.

So, as you may expect, I'm delighted that they've put this book into production, and I will be recommending it willy-nilly over the next few years. (I've already been using many bits from its drafts in my talks.) However, I do want to put in a few notes of caution. They do a good job of explaining why their approach to surveys makes them a good basis for their data. However, they are still surveys that capture subjective perceptions, and I wonder how their population sample reflects the general IT world. I'll have more confidence in their results when other teams, using different approaches, are able to confirm their reasoning. The book already has some of this, as the work done by Google on team cultures provides further evidence to support their judgment on how important a Westrum-generative organizational culture is for effective software teams. Such further work would also make me less concerned that their conclusions confirm much of my advocacy—confirmation bias is a strong force (although I mostly notice it in others ;-)). We should also remember that their book focuses on IT delivery, that is, the journey from commit to production, not the entire software development process.

But these quibbles, while present, shouldn't distract us from the main thrust of this book. These surveys, and the careful analysis done on them, provide some of the best justification around for practices that can significantly improve most IT organizations. Anyone running an IT group should take a good hard look at these techniques and work to use them to improve their practice. Anyone working with an IT group, either internally or from an IT delivery company like ours, should look for these practices in place and a steady program of continuous improvement to go with them. Forsgren, Humble, and Kim have laid out a picture of what effective IT looks like in 2017, and IT practitioners should be using this as a map to join the high performers.

Martin Fowler
Chief Scientist, ThoughtWorks

FOREWORD

By Courtney Kissler

My journey started in the summer of 2011. I was working at Nordstrom and we had made a strategic decision to focus on digital as the growth engine. Up until that point, our IT organization was optimized for cost; I shared in my DevOps Enterprise Summit 2014 presentation that one of my "aha" moments was the shift to optimizing for speed. I made a lot of mistakes along the way and wish I had access to the information in this book back then. Common traps were stepped in—like trying a top-down mandate to adopt Agile, thinking it was one size fits all, not focusing on measurement (or the right things to measure), leadership behavior not changing, and treating the transformation like a program instead of creating a learning organization (never done).

Throughout the journey, the focus was moving to outcome-based team structures, knowing our cycle time (by understanding our value stream map), limiting the blast radius (starting with one to two teams vs. boiling the ocean), using data to drive actions and decisions, acknowledging that work is work (don't have a backlog of features and a backlog of technical debt and a backlog of operational work; instead, have a single backlog because NFRs are features and reducing technical debt improves stability of the product). None of this happened overnight, and it took a lot of experimentation and adjusting along the way.

What I know to be true based on my experience is that adopting the guidance in this book *will* make your organization higher performing. It works for all types of software delivery and is methodology agnostic. I have personally experienced it and have multiple examples of applying these practices within mainframe environments, traditional packaged software application delivery teams, and product teams. It can work across the board. It takes discipline, persistence, transformational leadership, and a focus on people. After all, people are an organization's #1 asset, but so often that is not how organizations operate. Even though the journey will not be easy, I can say that it is definitely worth it, and not only will you see better results, your team will be happier. As an example, when we started measuring eNPS, the teams practicing these techniques had the highest scores throughout our technology organization.

Another thing I learned along the way is how critical it is to have senior leadership support. And support in actions, not words. Senior leaders need to demonstrate their commitment to creating a learning organization. I will share the behaviors I try to model with my teams. I believe passionately in honoring and extracting reality. If I am a senior leader and my team doesn't feel comfortable sharing risks, then I will never truly know reality. And, if I'm not genuinely curious and only show up when there's a failure, then I am failing as a senior leader. It's important to build trust and to demonstrate that failure leads to inquiry (see the Westrum model in this book).

You will encounter skeptics along the way. I heard things like "DevOps is the new Agile," "Lean doesn't apply to software delivery," "Of course this worked for the mobile app team. They are a unicorn." When I encountered the skeptics, I attempted to use external examples to influence the discussion. I leveraged mentors along the way—without them, it would have been challenging to stay focused. Having the information in this book would have been extremely helpful and I strongly encourage you to use it within your organization. I have spent most of my career in retail; in that industry, it has become more and more critical to evolve, and shipping software is now part of the DNA of every organization. Don't ignore the science outlined in this book. It will help you accelerate your transformation to a high-performing technology organization.

Courtney Kissler
VP Digital Platform Engineering, Nike

QUICK REFERENCE: CAPABILITIES TO DRIVE IMPROVEMENT

Our research has uncovered 24 key capabilities that drive improvements in software delivery performance. This reference will point you to them in the book. A detailed guide is found in Appendix A. They are presented in no particular order.

The capabilities are classified into five categories:

- Continuous delivery
- Architecture
- Product and process
- Lean management and monitoring
- Cultural

CONTINUOUS DELIVERY CAPABILITIES

1. Version control: Chapter 4
2. Deployment automation: Chapter 4
3. Continuous integration: Chapter 4
4. Trunk-based development: Chapter 4
5. Test automation: Chapter 4
6. Test data management: Chapter 4
7. Shift left on security: Chapter 6
8. Continuous delivery (CD): Chapter 4

ARCHITECTURE CAPABILITIES

9. Loosely coupled architecture: Chapter 5
10. Empowered teams: Chapter 5

PRODUCT AND PROCESS CAPABILITIES

11. Customer feedback: Chapter 8
12. Value stream: Chapter 8
13. Working in small batches: Chapter 8
14. Team experimentation: Chapter 8

LEAN MANAGEMENT AND MONITORING CAPABILITIES

15. Change approval processes: Chapter 7
16. Monitoring: Chapter 7
17. Proactive notification: Chapter 13
18. WIP limits: Chapter 7
19. Visualizing work: Chapter 7

CULTURAL CAPABILITIES

20. Westrum organizational culture: Chapter 3
21. Supporting learning: Chapter 10
22. Collaboration among teams: Chapters 3 and 5
23. Job satisfaction: Chapter 10
24. Transformational leadership: Chapter 11

PREFACE

Beginning in late 2013, we embarked on a four-year research journey to investigate what capabilities and practices are important to accelerate the development and delivery of software and, in turn, value to companies. These results are seen in their profitability, productivity, and market share. We see similarly strong effects in noncommercial outcomes of effectiveness, efficiency, and customer satisfaction.

This research fills a need that isn't currently served in the market. By using rigorous research methods traditionally only found in academia, and making it accessible to industry, our goal is to advance the state of software development and delivery. By helping the industry identify and understand the capabilities that actually drive performance improvements in a statistically meaningful way—more than just anecdote, and beyond the experiences of one or a few teams—we can help the industry improve.

To conduct the research found in this book (in addition to research we still actively conduct), we use cross-sectional studies. The same methods are used in healthcare research (e.g., to investigate the relationship between beer and obesity, Bobak et al. 2003), workplace research (e.g., to study the relationship between the work environment and cardiovascular disease, Johnson and Hall 1988), and memory research (e.g., to investigate differences in development and decline in memory, Alloway and Alloway 2013). As we want to truly investigate the industry and understand what drives improvement in software and organizational performance in a meaningful way, we use rigorous academic research design methods and publish much of our work in academic peer-reviewed journals. For more information about the methods used in our research, check out Part II: The Research.

THE RESEARCH

Our research collected over 23,000 survey responses from around the world. We heard from over 2,000 unique organizations, from small startups of under five employees to large enterprises with over 10,000 employees. We collected data from startups and cutting-edge internet companies as well as highly regulated industries, such as finance, healthcare, and government. Our data and analysis includes software developed on brand new "greenfield" platforms as well as legacy code maintenance and development.

The findings in this book will apply whether you're using a traditional "waterfall" methodology (also known as gated, structured, or plan-driven) and just beginning your technology transformation, or whether you have been implementing Agile and DevOps practices for years. This is true because software delivery is an exercise in continuous improvement, and our research shows that year over year the best keep getting better, and those who fail to improve fall further and further behind.

Improvement Is Possible for Everyone

Our quest to understand how to measure and improve software delivery was full of insights and surprises. The moral of the story, borne out in the data, is this: improvements in software delivery are possible for every team and in every company, as long as leadership provides consistent support—

including time, actions, and resources—demonstrating a true commitment to improvement, and as long as team members commit themselves to the work.

Our goal in writing this book is to share what we have learned so that we can help organizations excel, grow happier teams who deliver better software faster, and help individuals and organizations thrive. The rest of this preface briefly describes the research, how it began, and how it was conducted. More detail about the science behind the study can be found in Part II of this book.

THE JOURNEY AND THE DATA

We are often asked about the genesis story of this research. It is based on a compelling curiosity for what makes high-performing technology organizations great, and how software makes organizations better. Each author spent time on parallel paths working to understand superior technical performance before joining forces in late 2013:

- **Nicole Forsgren** has a PhD in Management Information Systems. Prior to 2013, she spent several years researching the factors that make technology impactful in organizations, particularly among the professionals that make software and support infrastructure. She has authored dozens of peer-reviewed articles on the subject. Before her PhD, she was a software and hardware engineer and a sysadmin.
- **Jez Humble** is the coauthor of *Continuous Delivery*, *Lean Enterprise*, and *The DevOps Handbook*. His first job after college was working at a startup in London in 2000, and

then from 2005–2015 he spent a decade at ThoughtWorks delivering software products and consulting as an infrastructure specialist, developer, and product manager.

- **Gene Kim** has been studying high-performing technology organizations since 1999. He was the founder and CTO of Tripwire for thirteen years and is the coauthor of many books, including *The Phoenix Project* and *The Visible Ops Handbook*.

In late 2013, Nicole, Jez, and Gene started working together with the team at Puppet in preparation for the *2014 State of DevOps Report*.[1] By combining practical expertise and academic rigor, the team was able to generate something unique in the industry: a report containing insights into how to help technology deliver value to employees, organizations, and customers in predictive ways. Over the next four reports, Nicole, Jez, and Gene continued collaborating with the Puppet team to iterate on research design and continuously improve the industry's understanding of what contributes to great software delivery, what enables great technical teams, and how companies can become high-performing organizations and win in the market by leveraging technology. This book covers four years of research findings, starting with that report (2014 through 2017).

[1] It is important to note that the *State of DevOps Report* got its start prior to 2014. In 2012, the team at Puppet Inc. invited Gene to participate in the second iteration of a study it was developing to better understand a little known phenomenon called DevOps, how it was being adopted, and the performance advantages organizations were seeing. Puppet had been a big proponent and driver of the movement as the idea of "DevOps" began to take shape following the first DevOpsDays, discussions on Twitter, and a seminal talk by John Allspaw and Paul Hammond. Gene then invited Jez to join the study, and together they collected and analyzed 4,000 survey responses from around the world, making it the largest survey of its kind.

To collect the data, each year we emailed invitations to our mailing lists and leveraged social media, including Twitter, LinkedIn, and Facebook. Our invitations targeted professionals working in technology, especially those familiar with software development and delivery paradigms and DevOps. We encouraged our readers to invite friends and peers who might also work in software development and delivery to help us broaden our reach. This is called snowball sampling, and we talk about why this was an appropriate data collection method for this research project in Chapter 15, "The Data for the Project."

The data for our project came from surveys. We used surveys because they are the best way to collect a large amount of data from thousands of organizations in a short amount of time. For a detailed discussion of why good research can be conducted from surveys, as well as the steps we took to ensure the data we collected was trustworthy and accurate, see Part II which covers the science and research behind the book.

Here is a brief outline of the research and how it evolved over the years.

2014: LAYING THE FOUNDATION. DELIVERY PERFORMANCE AND ORGANIZATIONAL PERFORMANCE

Our research goals for the first year were to lay a foundation for understanding software development and delivery in organizations. Some key research questions were:

- What does it mean to deliver software, and can it be measured?
- Does software delivery impact organizations?

- Does culture matter, and how do we measure it?
- What technical practices appear to be important?

We were pleasantly surprised by many of the results in the first year. We discovered that software development and delivery can be measured in a statistically meaningful way, and that high performers do it in consistently good ways that are significantly better than many other companies. We also found that throughput and stability move together, and that an organization's ability to make software positively impacts profitability, productivity, and market share. We saw that culture and technical practices matter, and found how to measure them. This is covered in Part I of this book.

The team also revised the way most of the data had been measured in the past, moving from simple yes/no questions to Likert-type questions (in which respondents choose from a range of options from "Strongly Disagree" to "Strongly Agree"). This simple change in survey questions let the team collect more nuanced data—shades of gray instead of black and white. This allowed for more detailed analysis. For a discussion of the authors' choice to use surveys for this research project and why you can trust their survey-based data, see Chapter 14, "Why Use a Survey."

2015: EXTENDING THE WORK AND DEEPENING THE ANALYSIS

Much like technology transformations and business growth, conducting research is all about iteration, incremental improvements, and revalidation of important results. Armed with our findings from the first year, our goals in year two were to revalidate and confirm some key findings (e.g., software delivery can be defined and measured in a statistically meaningful way, software delivery impacts organizational performance) while also extending the model.

These were some of the research questions:

- Can we revalidate that software delivery impacts organizational performance?
- Do technical practices and automation impact software delivery?
- Do lean management practices impact software delivery?
- Do technical practices and Lean management practices impact aspects of work that affect our workforce—such as anxiety associated with code deployments and burnout?

Once again, we got some great confirmations and some surprises. Our hypotheses were supported, confirming and extending the work we had done the previous year. These results can be found in Part I.

2016: EXPANDING OUR LOOK INTO TECHNICAL PRACTICES AND EXPLORING THE FUZZY FRONT END

In year three, we again built on the core foundation of our model and extended it to explore the significance of additional technical practices (such as security, trunk-based development, and test data management). Inspired by conversations with colleagues working in product management, we also extended our investigation further upstream, to see if we could measure the impact of the current move away from traditional project management practices to applying Lean principles in product management. We extended our investigation to include quality measures such as defects, rework, and security remediation. Finally, we included additional questions to help us understand how technical practices influence human capital: employee Net Promoter Score (eNPS) and work identity—a factor that is likely to decrease burnout.

These were our research questions:

- Does the integration of security into software development and delivery help the process or slow it down?
- Does trunk-based development contribute to better software delivery?
- Is a Lean approach to product management an important aspect of software development and delivery?
- Do good technical practices contribute to strong company loyalty?

2017: INCLUDING ARCHITECTURE, EXPLORING THE ROLE OF LEADERS, AND MEASURING SUCCESS IN NOT-FOR-PROFIT ORGANIZATIONS

Year four of the research saw us moving into questions about how systems are architected and the impact architecture has on teams' and organizations' ability to deliver software and value. We also extended our research to include measures of value that extended beyond profitability, productivity, and market share, allowing the analysis to speak to a not-for-profit audience. The research this year also explored the role of leaders to measure the impact of transformational leadership in organizations.

Our driving research questions in year four were:

- What architectural practices drive improvements in software delivery performance?
- How does transformational leadership impact software delivery?
- Does software delivery impact not-for-profit outcomes?

CONCLUSION

We hope that as you read this book you discover, as a technologist and technology leader, the essential components to making your organization better—starting with software delivery. It is through improving our ability to deliver software that organizations can deliver features faster, pivot when needed, respond to compliance and security changes, and take advantage of fast feedback to attract new customers and delight existing ones.

In the chapters that follow, we identify the key capabilities that drive the software delivery performance (and define what software delivery performance is) and briefly touch on the key points in each. Part I of the book presents our findings, Part II discusses the science and research behind our results, and finally, Part III presents a case study of what is possible when organizations adopt and implement these capabilities in order to drive performance.

PART ONE
WHAT WE FOUND

Armed with robust data-gathering and statistical analysis techniques (discussed in detail in Part II), we have been able to discover significant and sometimes surprising results over the past several years working on the State of DevOps Report. We've been able to measure and quantify software delivery performance, its impact on organizational performance, and the various capabilities that contribute to these outcomes.

These capabilities fall into various categories—such as technical, process, and cultural. We've measured the impact of technical practices on culture, and the effect of culture on delivery and organizational performance. For capabilities as disparate as architecture and product management, we've looked at their contribution to these and other important sustainability outcomes such as burnout and deployment pain.

In this part of the book we present our results.

ACCELERATE

"Business as usual" is no longer enough to remain competitive. Organizations in all industries, from finance and banking to retail, telecommunications, and even government, are turning away from delivering new products and services using big projects with long lead times. Instead, they are using small teams that work in short cycles and measure feedback from users to build products and services that delight their customers and rapidly deliver value to their organizations. These high performers are working incessantly to get better at what they do, letting no obstacles stand in their path, even in the face of high levels of risk and uncertainty about how they may achieve their goals.

To remain competitive and excel in the market, organizations must accelerate:

- delivery of goods and services to delight their customers;
- engagement with the market to detect and understand customer demand;
- anticipation of compliance and regulatory changes that impact their systems; and
- response to potential risks such as security threats or changes in the economy.

At the heart of this acceleration is software. This is true of organizations in any industry vertical. Banks no longer deliver value by holding gold bars in vaults but by trading faster and more securely, and by discovering new channels and products to engage customers. Retailers win and retain customers by offering them superior selection and service, with service coming in the form of a fast check-out experience, recommended goods at check-out, or a seamless online/offline shopping experience—all of which are enabled by technology. Government organizations cite the ability to harness technology as the key to serving the public more effectively and efficiently while being parsimonious with taxpayer dollars.

Software and technology are key differentiators for organizations to deliver value to customers and stakeholders. We've found it in our own research outlined in this book—and others have found it, too. For example, a recent study by James Bessen of Boston University found that the strategic use of technology explains revenue and productivity gains more than mergers and acquisitions (M&A) and entrepreneurship (2017). Andrew McAfee and Erik Brynjolfsson have also found a link between technology and profitability (2008).

Software is transforming and accelerating organizations of all kinds. The practices and capabilities we talk about in this book have emerged from what is now known as the DevOps movement, and they are transforming industries everywhere. DevOps emerged from a small number of organizations facing a wicked problem: how to build secure, resilient, rapidly evolving distributed systems at scale. In order to remain competitive, organizations must learn how to solve these problems. We see that large enterprises with long histories and decades-old technologies also gain significant benefits, such as accelerated delivery and lower costs, through adopting the capabilities we outline in this book.

Although many organizations have achieved great success with their technology transformations (notable examples include web-scale tech giants such as Netflix, Amazon, Google, and Facebook, as well as more traditional large organizations including Capital One, Target, and the US Federal Government's Technology Transformation Service and US Digital Service), there is still a lot of work to be done—both in the broader industry and within individual organizations. A recent Forrester (Stroud et al. 2017) report found that 31% of the industry is not using practices and principles that are widely considered to be necessary for accelerating technology transformations, such as continuous integration and continuous delivery, Lean practices, and a collaborative culture (i.e., capabilities championed by the DevOps movement). However, we also know that technology and software transformations are imperative in organizations today. A recent Gartner study found that 47% of CEOs face pressure from their board to digitally transform (Panetta 2017).

Within organizations, technology transformation journeys are at different stages, and reports suggest there is more work to be done than many of us currently believe. Another Forrester report states that DevOps is accelerating technology, but that organizations often overestimate their progress (Klavens et al. 2017). Furthermore, the report points out that executives are especially prone to overestimating their progress when compared to those who are actually doing the work.

These findings about the disconnect between executive and practitioner estimates of DevOps maturity highlight two considerations that are often missed by leaders. First, if we assume the estimates of DevOps maturity or capabilities from practitioners are more accurate—because they are closer to the work—the potential for value delivery and growth within organizations is much greater than executives currently realize. Second, the disconnect makes clear the need to measure DevOps capabilities accurately and to communicate these measurement results to leaders, who can use them to make decisions and inform strategy about their organization's technology posture.

FOCUS ON CAPABILITIES, NOT MATURITY

Technology leaders need to deliver software quickly and reliably to win in the market. For many companies, this requires significant changes to the way we deliver software. The key to successful change is measuring and understanding the right things with a focus on capabilities—not on maturity.

While maturity models are very popular in the industry, we cannot stress enough that maturity models are not the appropriate tool to use or mindset to have. Instead, shifting to a capabilities model of measurement is essential for organizations wanting to accelerate software delivery. This is due to four factors.

First, maturity models focus on helping an organization "arrive" at a mature state and then declare themselves done with their journey, whereas technology transformations should follow a continuous improvement paradigm. Alternatively, capability models focus on helping an organization continually improve and progress, realizing that the technological and business landscape is ever-changing. The most innovative companies and highest-performing organizations are always striving to be better and never consider themselves "mature" or "done" with their improvement or transformation journey—and we see this in our research.

Second, maturity models are quite often a "lock-step" or linear formula, prescribing a similar set of technologies, tooling, or capabilities for every set of teams and organizations to progress through. Maturity models assume that "Level 1" and "Level 2" look the same across all teams and organizations, but those of us who work in technology know this is not the case. In contrast, capability models are multidimensional and dynamic, allowing different parts of the organization to take a customized approach to improvement, and focus on capabilities that will give them the most benefit based on their current context and their short- and long-term goals. Teams have their own context, their own systems, their own goals, and their own constraints, and what we should focus on next to accelerate our transformation depends on those things.

Third, capability models focus on key outcomes and how the capabilities, or levers, drive improvement in those outcomes—that is, they are outcome based. This provides technical leadership with clear direction and strategy on high-level goals (with a focus on capabilities to improve key outcomes). It also enables team leaders and individual contributors to set improvement goals related to the capabilities their team is focusing on for the current time period. Most maturity models simply measure the technical proficiency or tooling install base in an organization without tying it to outcomes. These end up being vanity metrics: while they can be relatively easy to measure, they don't tell us anything about the impact they have on the business.

Fourth, maturity models define a static level of technological, process, and organizational abilities to achieve. They do not take into account the ever-changing nature of the technology and business landscape. Our own research and data have confirmed that the industry is changing: what is good enough and even "high-performing" today is no longer good enough in the next year. In contrast, capability models allow for dynamically changing environments and allow teams and organizations to focus on developing the skills and capabilities needed to remain competitive.

By focusing on a capabilities paradigm, organizations can continuously drive improvement. And by focusing on the *right* capabilities, organizations can drive improvements in their outcomes, allowing them to develop and deliver software with improved speed and stability. In fact, we see that the highest performers do exactly this, continually reaching for gains year over year and never settling for yesterday's accomplishments.

EVIDENCE-BASED TRANSFORMATIONS FOCUS ON KEY CAPABILITIES

Within both capability and maturity model frameworks, there are disagreements about *which* capabilities to focus on. Product vendors often favor capabilities that align with their product offerings. Consultants favor capabilities that align with their background, their offering, and their homegrown assessment tool. We have seen organizations try to design their own assessment models, choose solutions that align with the skill sets of internal champions, or succumb to analysis paralysis because of the sheer number of areas that need improvement in their organization.

A more guided, evidence-based solution is needed, and the approach discussed in this book describes such a solution.

Our research has yielded insights into what enables both software delivery performance and organizational performance as seen in profitability, productivity, and market share. In fact, our research shows that none of the following often-cited factors predicted performance:

- age and technology used for the application (for example, mainframe "systems of record" vs. greenfield "systems of engagement")

- whether operations teams or development teams performed deployments
- whether a change approval board (CAB) is implemented

The things that *do* make a difference in the success of software delivery and organizational performance are those that the highest performers and most innovative companies use to get ahead. Our research has identified 24 key capabilities that drive improvement in software delivery performance and, in turn, organizational performance. These capabilities are easy to define, measure, and improve.[1] This book will get you started on defining and measuring these capabilities. We will also point you to some fantastic resources for improving them, so you can accelerate your own technology transformation journey.

THE VALUE OF ADOPTING DEVOPS

You may be asking yourself: How do we know that these capabilities are drivers of technology and organizational performance, and why can we say it with such confidence?

The findings from our research program show clearly that the value of adopting DevOps is even larger than we had initially thought, and the gap between high and low performers continues to grow.

We discuss how we measure software delivery performance and how our cohort performs in detail in the following chapter. To summarize, in 2017 we found that, when compared to low performers, the high performers have:

[1] These 24 capabilities are listed, along with a pointer to the chapter that discusses them, in Appendix A.

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

When compared to the 2016 results, the gap between high performers and low performers narrowed for tempo (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 tempo but not investing enough in building quality into the process. The result is larger deployment failures that take 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.

You may be wondering: How do high-performing teams achieve such amazing software delivery performance? They do this by turning the right levers—that is, by improving the right capabilities.

Over our four-year research program we have been able to identify the capabilities that drive performance in software delivery and impact organizational performance, and we have found that they work for all types of organizations. Our research investigated organizations of all sizes, in all industries, using legacy and greenfield technology stacks around the world—so the findings in this book will apply to the teams in your organization too.

CHAPTER 2

MEASURING PERFORMANCE

There are many frameworks and methodologies that aim to improve the way we build software products and services. We wanted to discover what works and what doesn't in a scientific way, starting with a definition of what "good" means in this context. This chapter presents the framework and methods we used to work towards this goal, and in particular the key outcome measures applied throughout the rest of this book.

By the end of this chapter, we hope you'll know enough about our approach to feel confident in the results we present in the rest of the book.

Measuring performance in the domain of software is hard—in part because, unlike manufacturing, the inventory is invisible. Furthermore, the way we break down work is relatively arbitrary, and the design and delivery activities—particularly in the Agile software development paradigm—happen simultaneously. Indeed, it's expected that we will change and evolve our design based on what we learn by trying to implement it. So our first step must be to define a valid, reliable measure of software delivery performance.

THE FLAWS IN PREVIOUS ATTEMPTS TO MEASURE PERFORMANCE

There have been many attempts to measure the performance of software teams. Most of these measurements focus on productivity. In general, they suffer from two drawbacks. First, they focus on *outputs* rather than *outcomes*. Second, they focus on individual or local measures rather than team or global ones. Let's take three examples: lines of code, velocity, and utilization.

Measuring productivity in terms of lines of code has a long history in software. Some companies even required developers to record the lines of code committed per week.[1] However, in reality we would prefer a 10-line solution to a 1,000-line solution to a problem. Rewarding developers for writing lines of code leads to bloated software that incurs higher maintenance costs and higher cost of change. Ideally, we should reward developers for solving business problems with the minimum amount of code—and it's even better if we can solve a problem without writing code at all or by deleting code (perhaps by a business process change). However, minimizing lines of code isn't an ideal measure either. At the extreme, this too has its drawbacks: accomplishing a task in a single line of code that no one else can understand is less desirable than writing a few lines of code that are easily understood and maintained.

With the advent of Agile software development came a new way to measure productivity: velocity. In many schools of Agile, problems are broken down into stories. Stories are then estimated by developers and assigned a number of "points" representing the relative effort expected to complete them. At the end of an

[1] There's a good story about how the Apple Lisa team's management discovered that lines of code were meaningless as a productivity metric: http://www.folklore.org/StoryView.py?story=Negative_2000_Lines_Of_Code.txt.

iteration, the total number of points signed off by the customer is recorded—this is the team's velocity. Velocity is designed to be used as a *capacity planning tool*; for example, it can be used to extrapolate how long it will take the team to complete all the work that has been planned and estimated. However, some managers have also used it as a way to measure team productivity, or even to compare teams.

Using velocity as a productivity metric has several flaws. First, velocity is a relative and team-dependent measure, not an absolute one. Teams usually have significantly different contexts which render their velocities incommensurable. Second, when velocity is used as a productivity measure, teams inevitably work to game their velocity. They inflate their estimates and focus on completing as many stories as possible at the expense of collaboration with other teams (which might decrease their velocity and increase the other team's velocity, making them look bad). Not only does this destroy the utility of velocity for its intended purpose, it also inhibits collaboration between teams.

Finally, many organizations measure utilization as a proxy for productivity. The problem with this method is that high utilization is only good up to a point. Once utilization gets above a certain level, there is no spare capacity (or "slack") to absorb unplanned work, changes to the plan, or improvement work. This results in longer lead times to complete work. Queue theory in math tells us that as utilization approaches 100%, lead times approach infinity—in other words, once you get to very high levels of utilization, it takes teams exponentially longer to get anything done. Since lead time—a measure of how fast work can be completed—is a productivity metric that doesn't suffer from the drawbacks of the other metrics we've seen, it's essential that we manage utilization to balance it against lead time in an economically optimal way.

MEASURING SOFTWARE DELIVERY PERFORMANCE

A successful measure of performance should have two key characteristics. First, it should focus on a global outcome to ensure teams aren't pitted against each other. The classic example is rewarding developers for throughput and operations for stability: this is a key contributor to the "wall of confusion" in which development throws poor quality code over the wall to operations, and operations puts in place painful change management processes as a way to inhibit change. Second, our measure should focus on outcomes not output: it shouldn't reward people for putting in large amounts of busywork that doesn't actually help achieve organizational goals.

In our search for measures of delivery performance that meet these criteria, we settled on four: delivery lead time, deployment frequency, time to restore service, and change fail rate. In this section, we'll discuss why we picked these particular measures.

The elevation of lead time as a metric is a key element of Lean theory. Lead time is the time it takes to go from a customer making a request to the request being satisfied. However, in the context of product development, where we aim to satisfy multiple customers in ways they may not anticipate, there are two parts to lead time: the time it takes to design and validate a product or feature, and the time to deliver the feature to customers. In the design part of the lead time, it's often unclear when to start the clock, and often there is high variability. For this reason, Reinertsen calls this part of the lead time the "fuzzy front end" (Reinertsen 2009). However, the *delivery* part of the lead time—the time it takes for work to be implemented, tested, and delivered—is easier to measure and has a lower variability. Table 2.1 (Kim et al. 2016) shows the distinction between these two domains.

Table 2.1 Design vs. Delivery

Product Design and Development	Product Delivery (Build, Testing, Deployment)
Create new products and services that solve customer problems using hypothesis-driven delivery, modern UX, design thinking.	Enable fast flow from development to production and reliable releases by standardizing work, and reducing variability and batch sizes.
Feature design and implementation may require work that has never been performed before.	Integration, test, and deployment must be performed continuously as quickly as possible.
Estimates are highly uncertain.	Cycle times should be well-known and predictable.
Outcomes are highly variable.	Outcomes should have low variability.

Shorter product delivery lead times are better since they enable faster feedback on what we are building and allow us to course correct more rapidly. Short lead times are also important when there is a defect or outage and we need to deliver a fix rapidly and with high confidence. We measured product delivery lead time as the time it takes to go from code committed to code successfully running in production, and asked survey respondents to choose from one of the following options:

- less than one hour
- less than one day
- between one day and one week
- between one week and one month
- between one month and six months
- more than six months

The second metric to consider is batch size. Reducing batch size is another central element of the Lean paradigm—indeed, it was one of the keys to the success of the Toyota production system. Reducing batch sizes reduces cycle times and variability in flow, accelerates feedback, reduces risk and overhead, improves efficiency, increases motivation and urgency, and reduces costs and schedule growth (Reinertsen 2009, Chapter 5). However, in software, batch size is hard to measure and communicate across contexts as there is no visible inventory. Therefore, we settled on deployment frequency as a proxy for batch size since it is easy to measure and typically has low variability.[2] By "deployment" we mean a software deployment to production or to an app store. A release (the changes that get deployed) will typically consist of multiple version control commits, unless the organization has achieved a single-piece flow where each commit can be released to production (a practice known as continuous deployment). We asked survey respondents how often their organization deploys code for the primary service or application they work on, offering the following options:

- on demand (multiple deploys per day)
- between once per hour and once per day
- between once per day and once per week
- between once per week and once per month
- between once per month and once every six months
- fewer than once every six months

[2] Strictly, deployment frequency is the reciprocal of batch size—the more frequently we deploy, the smaller the size of the batch. For more on measuring batch size in the context of IT service management, see Forsgren and Humble (2016).

Delivery lead times and deployment frequency are both measures of software delivery performance *tempo*. However, we wanted to investigate whether teams who improved their performance were doing so at the expense of the stability of the systems they were working on. Traditionally, reliability is measured as time between failures. However, in modern software products and services, which are rapidly changing complex systems, failure is inevitable, so the key question becomes: How quickly can service be restored? We asked respondents how long it generally takes to restore service for the primary application or service they work on when a service incident (e.g., unplanned outage, service impairment) occurs, offering the same options as for lead time (above).

Finally, a key metric when making changes to systems is what percentage of changes to production (including, for example, software releases and infrastructure configuration changes) fail. In the context of Lean, this is the same as percent complete and accurate for the product delivery process, and is a key quality metric. We asked respondents what percentage of changes for the primary application or service they work on either result in degraded service or subsequently require remediation (e.g., lead to service impairment or outage, require a hotfix, a rollback, a fix-forward, or a patch). The four measures selected are shown in Figure 2.1.

Software Delivery Performance
Lead Time
Deployment Frequency
Mean Time to Restore (MTTR)
Change Fail Percentage

Figure 2.1: Software Delivery Performance

In order to analyze delivery performance across the cohort we surveyed, we used a technique called *cluster analysis*. Cluster analysis is a foundational technique in statistical data analysis that attempts to group responses so that responses in the same group are more similar to each other than to responses in other groups. Each measurement is put on a separate dimension, and the clustering algorithm attempts to minimize the distance between all cluster members and maximize differences between clusters. This technique has no understanding of the semantics of responses—in other words, it doesn't know what counts as a "good" or "bad" response for any of the measures.[3]

This data-driven approach that categorizes the data without any bias toward "good" or "bad" gives us an opportunity to view trends in the industry without biasing the results a priori. Using cluster analysis also allowed us to identify categories of software delivery performance seen in the industry: Are there high performers and low performers, and what characteristics do they have?

We applied cluster analysis in all four years of the research project and found that every year, there were significantly different categories of software delivery performance in the industry. We also found that all four measures of software delivery performance are good classifiers and that the groups we identified in the analysis—high, medium, and low performers—were all significantly different across all four measures.

Tables 2.2 and 2.3 show you the details for software delivery performance for the last two years of our research (2016 and 2017).

[3] For more on cluster analysis, see Appendix B.

Table 2.2 Software Delivery Performance for 2016

2016	High Performers	Medium Performers	Low Performers
Deployment Frequency	On demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every six months
Lead Time for Changes	Less than one hour	Between one week and one month	Between one month and six months
MTTR	Less than one hour	Less than one day	Less than one day*
Change Failure Rate	0–15%	31–45%	16–30%

Table 2.3 Software Delivery Performance for 2017

2017	High Performers	Medium Performers	Low Performers
Deployment Frequency	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	Less than one hour	Between one week and one month	Between one week and one month*
MTTR	Less than one hour	Less than one day	Between one day and one week
Change Failure Rate	0–15%	0–15%	31–45%

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

Astonishingly, these results demonstrate that there is no trade-off between improving performance and achieving higher levels of stability and quality. Rather, high performers do better at *all* of these measures. This is precisely what the Agile and Lean movements predict, but much dogma in our industry still rests on the false assumption that moving faster means *trading off* against other performance goals, rather than enabling and reinforcing them.[4]

Furthermore, over the last few years we've found that the high-performing cluster is pulling away from the pack. The DevOps mantra of continuous improvement is both exciting and real, pushing companies to be their best, and leaving behind those who do not improve. Clearly, what was state of the art three years ago is just not good enough for today's business environment.

Compared to 2016, high performers in 2017 maintained or improved their performance, consistently maximizing both tempo and stability. Low performers, on the other hand, maintained the same level of throughput from 2014–2016 and only started to increase in 2017—likely realizing that the rest of the industry was pulling away from them. In 2017, we saw low performers lose some ground in stability. We suspect this is due to attempts to increase tempo ("work harder!") which fail to address the underlying obstacles to improved overall performance (for example, rearchitecture, process improvement, and automation). We show the trends in Figures 2.2 and 2.3.

[4] See https://continuousdelivery.com/2016/04/the-flaw-at-the-heart-of-bimodal-it/ for an analysis of problems with the bimodal approach to ITSM, which rests on this false assumption.

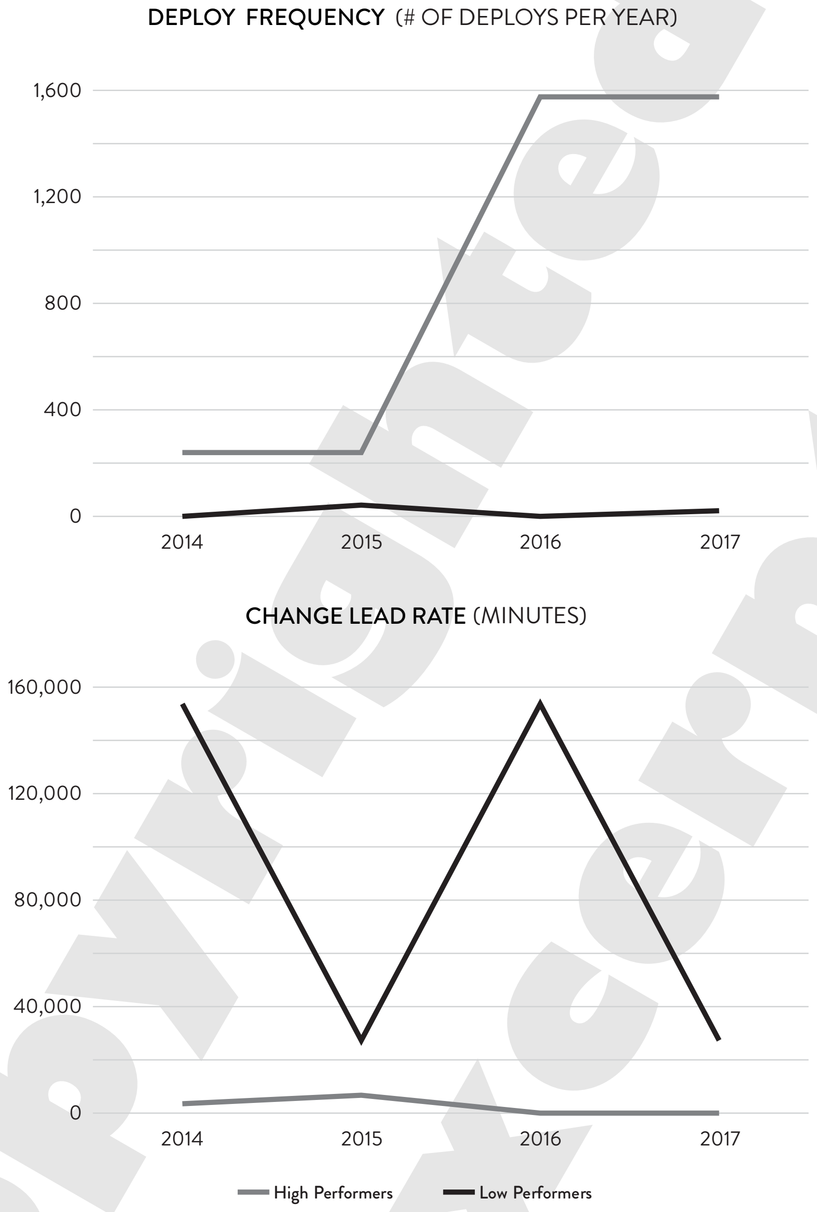


Figure 2.2: Year over Year Trends: Tempo

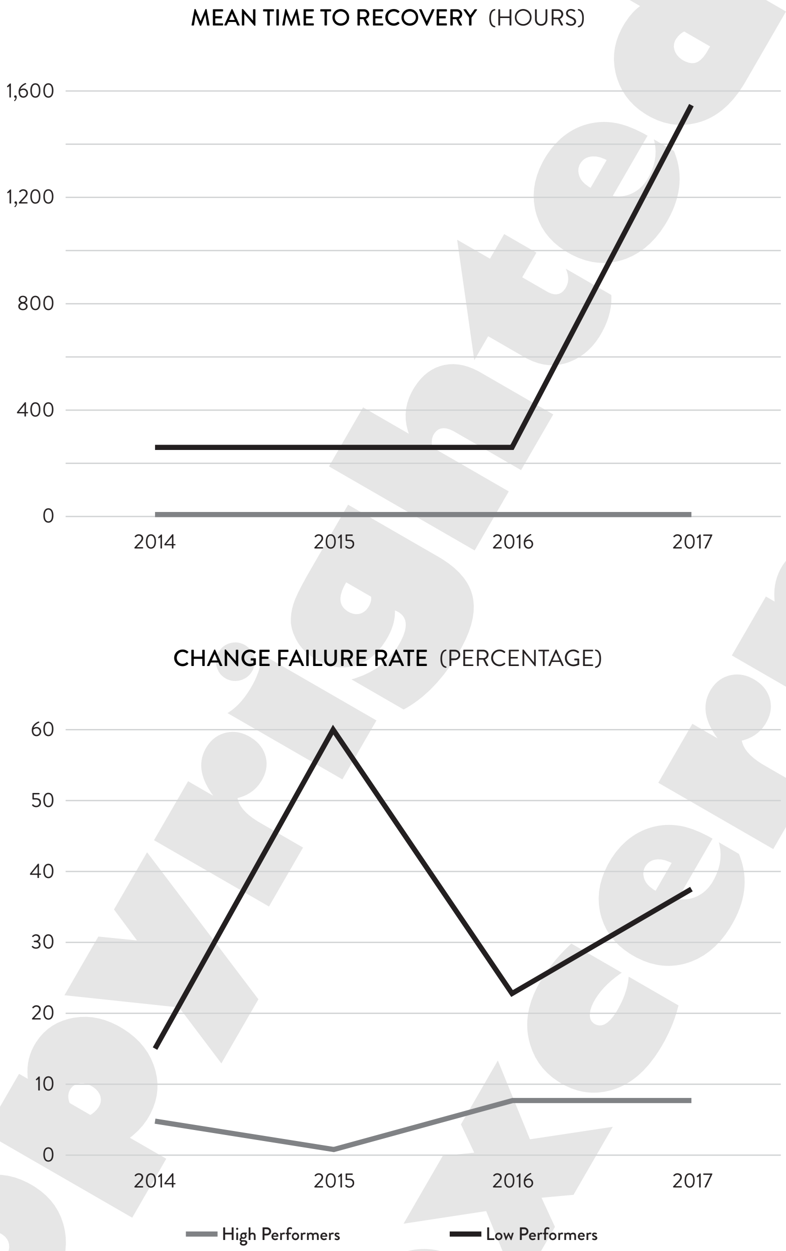


Figure 2.3: Year over Year Trends: Stability

Surprise!

Observant readers will notice that medium performers do worse than low performers on change fail rate in 2016. 2016 is the first year of our research where we see slightly inconsistent performance across our measures in any of our performance groups, and we see it in medium and low performers. Our research doesn't conclusively explain this, but we have a few ideas about why this might be the case.

One possible explanation is that medium performers are working along their technology transformation journey and dealing with the challenges that come from large-scale rearchitecture work, such as transitioning legacy code bases. This would also match another piece of the data from the 2016 study, where we found that medium performers spend more time on unplanned rework than low performers—because they report spending a greater proportion of time on new work.

We believe this new work could be occurring at the expense of ignoring critical rework, thus racking up technical debt which in turn leads to more fragile systems and, therefore, a higher change fail rate.

We have found a valid, reliable way to measure software delivery performance that satisfies the requirements we laid out. It focuses on global, system-level goals, and measures outcomes that different functions must collaborate in order to improve. The next question we wanted to answer is: Does software delivery performance matter?

THE IMPACT OF DELIVERY PERFORMANCE ON ORGANIZATIONAL PERFORMANCE

In order to measure organizational performance, survey respondents were asked to rate their organization's relative performance across several dimensions: profitability, market share, and productivity. This is a scale that has been validated multiple times in prior research (Widener 2007). This measure of organizational performance has also been found to be highly correlated to measures of return on investment (ROI), and it is robust to economic cycles—a great measure for our purposes. Analysis over several years shows that high-performing organizations were consistently *twice as likely* to exceed these goals as low performers. This demonstrates that your organization's software delivery capability can in fact provide a competitive advantage to your business.

In 2017, our research also explored how IT performance affects an organization's ability to achieve broader organizational goals—that is, goals that go beyond simple profit and revenue measures. Whether you're trying to generate profits or not, any organization today depends on technology to achieve its mission and provide value to its customers or stakeholders quickly, reliably, and securely. Whatever the mission, how a technology organization performs can predict overall organizational performance. To measure noncommercial goals, we used a scale that has been validated multiple times and is particularly well-suited for this purpose (Cavalluzzo and Ittner 2004). We found that high performers were also twice as likely to exceed objectives in quantity of goods and services, operating efficiency, customer satisfaction, quality of products or services, and achieving organization or mission goals. We show this relationship in Figure 2.4.

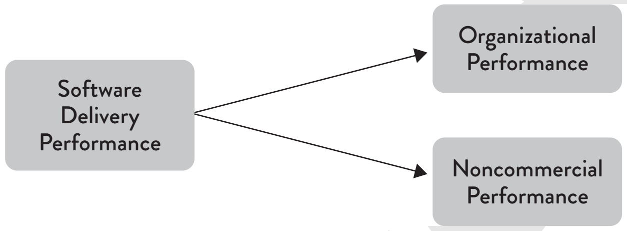

Software Delivery Performance
Organizational Performance
Noncommercial Performance

Figure 2.4: Impacts of Software Delivery Performance

Reading the Figures in This Book

We will include figures to help guide you through the research.

- When you see a box, this is a construct we have measured. (For details on constructs, see Chapter 13.)
- When you see an arrow linking boxes, this signifies a predictive relationship. You read that right: the research in this book includes analyses that go beyond correlation into prediction. (For details, see Chapter 12 on inferential prediction.) You can read these arrows using the words "drives," "predicts," "affects," or "impacts." These are all positive relationships unless otherwise noted.

For example, Figure 2.4 could be read as "software delivery performance impacts organizational performance and noncommercial performance."

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.

The fact that software delivery performance matters provides a strong argument against outsourcing the development of software that is strategic to your business, and instead bringing this capability into the core of your organization. Even the US Federal Government, through initiatives such as the US Digital Service and its agency affiliates and the General Services Administration's Technology Transformation Service team, has invested in bringing software development capability in-house for strategic initiatives.

In contrast, most software used by businesses (such as office productivity software and payroll systems) are not strategic and should in many cases be acquired using the software-as-a-service model. Distinguishing which software is strategic and which isn't, and managing them appropriately, is of enormous importance. This topic is dealt with at length by Simon Wardley, creator of the Wardley mapping method (Wardley 2015).

DRIVING CHANGE

Now that we have defined software delivery performance in a way that is rigorous and measurable, we can make evidence-based decisions on how to improve the performance of teams building software-based products and services. We can compare and benchmark teams against the larger organizations they work in and against the wider industry. We can measure their improvement—or backsliding—over time. And perhaps most exciting of all, we can go beyond correlation and start testing prediction. We can test hypotheses about which practices—from managing work in process to test automation—actually impact delivery performance and the strength of these effects. We can measure other outcomes we care about, such as team burnout and deployment pain. We can answer questions like, “Do change management boards actually improve delivery performance?” (Spoiler alert: they do not; they are negatively correlated with tempo and stability.)

As we show in the next chapter, it is also possible to model and measure culture quantitatively. This enables us to measure the effect of DevOps and continuous delivery practices on culture and, in turn, the effect of culture on software delivery performance and organizational performance. Our ability to measure and reason about practices, culture, and outcomes is an incredibly powerful tool that can be used to great positive effect in the pursuit of ever higher performance.

You can, of course, use these tools to model your own performance. Use Table 2.3 to discover where in our taxonomy you fall. Use our measures for lead time, deployment frequency, time to restore service, and change fail rate, and ask your teams to set targets for these measures.

However, it is essential to use these tools carefully. In organizations with a learning culture, they are incredibly powerful. But “in pathological and bureaucratic organizational cultures, measurement is used as a form of control, and people hide information that challenges existing rules, strategies, and power structures. As Deming said, ‘whenever there is fear, you get the wrong numbers’” (Humble et al. 2014, p. 56). Before you are ready to deploy a scientific approach to improving performance, you must first understand and develop your culture. It is to this topic we now turn.

PART TWO THE RESEARCH

To establish what we presented in Part I, we had to go beyond case studies and stories and into rigorous research methods. This allowed us to identify the practices that are the strongest predictors of success for all organizations of any size in any industry.

In the first part of the book, we discussed the results of this research program and outlined why technology is a key value driver and differentiator for all organizations today. Now, we present the science behind the research findings in Part I.

CHAPTER 12

THE SCIENCE BEHIND THIS BOOK

Every day, our news feeds are full of strategies designed to make our lives easier, make us happier, and help us take over the world. We also hear stories about how teams and organizations use different strategies to transform their technology and win in the market. But how are we to know which actions we take just *happen* to correspond to the changes we observe in our environment and which actions are driving these changes? This is where rigorous primary research comes into play. But what do we mean by "rigorous" and "primary"?

PRIMARY AND SECONDARY RESEARCH

Research falls into two broad classes: primary and secondary research. The key difference between these two types is who collects the data. Secondary research utilizes data that was collected by someone else. Examples of secondary research you are probably familiar with are book reports or research reports we all completed in school or university: we collected existing information, summarized it, and (hopefully) added in our own insights about what was found. Common examples of this also include case studies and some market research reports. Secondary research reports can be valuable, particularly if the existing data is difficult to find, the summary is particularly insightful, or the reports are delivered at regular intervals. Secondary research is generally faster and less expensive to conduct, but the data may not be well suited to the research team because they are bound by whatever data already exists.

In contrast, primary research involves collecting new data by the research team. An example of primary research includes the United States Census. The research team collects new data every ten years to report on demographic and population statistics for the country. Primary research is valuable because it can report information that is not already known and provide insights that are not available in existing datasets. Primary research gives researchers more power and control over the questions they can address, though it is generally more costly and time intensive to conduct. This book and the State of DevOps Reports are based on primary research.

QUALITATIVE AND QUANTITATIVE RESEARCH

Research can be qualitative or quantitative. Qualitative research is any kind of research whose data isn't in numerical form. This can include interviews, blog posts, Twitter posts, long-form log data, and long-form observations from ethnographers. Many people assume that survey research is qualitative because it doesn't come from computer systems, but that isn't necessarily true; it depends on the kinds of questions asked in the survey. Qualitative data is very descriptive and can allow for more insights and emergent behavior to be discovered by researchers, particularly in complex or new areas. However, it is often more difficult and costly to analyze; efforts to analyze qualitative data using automated means often codify the data into a numerical format, making it quantitative.

Quantitative research is any kind of research with data that includes numbers. These can include system data (in numerical format) or stock data. System data is any data generated from our tools; one example is log data. It can also include survey data, if the survey asks questions that capture responses in numerical format—preferably on a scale. The research presented in this book is quantitative, because it was collected using a Likert-type survey instrument.

What Is a Likert-Type Scale?

A Likert-type scale records responses and assigns them a number value. For example, "Strongly disagree" would be given a value of 1, neutral a value of 4, and "Strongly agree" a value of 7. This provides a consistent approach to measurement across all research subjects, and provides a numerical base for researchers to use in their analysis.

TYPES OF ANALYSIS

Quantitative research allows us to do statistical data analysis. According to a framework presented by Dr. Jeffrey Leek at Johns Hopkins Bloomberg School of Public Health (Leek 2013), there are six types of data analysis (given below in the order of increasing complexity). This complexity is due to the knowledge required by the data scientist, the costs involved in the analysis, and the time required to perform the analysis. These levels of analysis are:

1. Descriptive
2. Exploratory
3. Inferential predictive
4. Predictive
5. Causal
6. Mechanistic

The analyses presented in this book fall into the first three categories of Dr. Leek's framework. We also describe an additional type of analysis, classification, which doesn't fit cleanly into the above framework.

DESCRIPTIVE ANALYSIS

Descriptive analysis is used in census reports. The data is summarized and reported—that is, described. This type of analysis takes the least amount of effort, and is often done as the first step of data analysis to help the research team understand their dataset (and, by extension, their sample and possibly population of users). In some cases, a report will stop at descriptive analysis, as in the case of population census reports.

What Is a Population and Sample, and Why Are They Important?

When talking about statistics and data analysis, the terms "population" and "sample" have special meanings. The *population* is the entire group of something you are interested in researching; this might be all of the people undergoing technology transformations, everyone who is a Site Reliability

Engineer at an organization, or even every line in a log file during a certain time period. A *sample* is a portion of that population that is carefully defined and selected. The sample is the dataset on which researchers perform their analyses. Sampling is used when the entire population is too big or not easily accessible for research. Careful and appropriate sampling methods are important to make sure the conclusions drawn from analyzing the sample are true for the population.

The most common example of descriptive analysis is the government census where population statistics are summarized and reported. Other examples include most vendor and analyst reports that collect data and report summary and aggregate statistics about the state of tool usage in an industry or the level of education and certification among technology professionals. The percentage of firms that have started their Agile or DevOps journeys as reported by Forrester (Klavens et al. 2017), the IDC report on average downtime cost (Elliot 2014), and the O'Reilly Data Science Salary Survey (King and Magoulas 2016) belong in this category.

These reports are very useful as a gauge of the current state of the industry, where reference groups (such as populations or industries) currently are, where they once were, and where the trends are pointing. However, descriptive findings are only as good as the underlying research design and data collection methods. Any reports that aim to represent the underlying population must be sure to sample that population carefully and discuss any limitations. A discussion of these considerations is beyond the scope of this book.

An example of descriptive analysis found in this book is the demographic information about our survey participants and the organizations they work in—what countries they come from, how large their organizations are, the industry vertical they work in, their job titles, and their gender (see Chapter 10).

EXPLORATORY ANALYSIS

Exploratory analysis is the next level of statistical analysis. This is a broad categorization that looks for relationships among the data and may include visualizations to identify patterns in the data. Outliers may also be detected in this step, though the researchers have to be careful to make sure that outliers are, in fact, outliers, and not legitimate members of the group.

Exploratory analyses are a fun and exciting part of the research process. For those who are divergent thinkers, this is often the stage where new ideas, new hypotheses, and new research projects are generated and proposed. Here, we discover how the variables in our data are related and we look for possible new connections and relationships. However, this should not be the end for a team that wants to make statements about prediction or causation.

Many people have heard the phrase "correlation doesn't imply causation," but what does that mean? The analyses done in the exploratory stage include correlation but not causation. Correlation looks at how closely two variables move together—or don't—but it doesn't tell us if one variable's movement predicts or causes the movement in another variable. Correlation analysis only tells us if two variables move in tandem or in opposition; it doesn't tell us why or what is causing it. Two variables moving together can always be due to a third variable or, sometimes, just chance.

A fantastic and fun set of examples that highlight high correlations due to chance can be found at the website Spurious Correlations.[1] The author Tyler Vigen has calculated examples of highly correlated variables that common sense tells us are not predictive and certainly not causal. For example, he shows (Figure 12.1) that the per capita cheese consumption is highly correlated with the number of people who died by becoming tangled in their bedsheets (with a correlation of 94.71% or $r = 0.9471$; see footnote 2 on correlations in this chapter). Surely cheese consumption doesn't cause strangulation by bedsheets. (And if it does—what kind of cheese?) It would be just as difficult to imagine strangulation by bedsheets causing cheese consumption—unless that is the food of choice at funerals and wakes around the country. (And again: What kind of cheese? That is a morbid marketing opportunity.) And yet, when we go "fishing in the data," our minds fill in the story

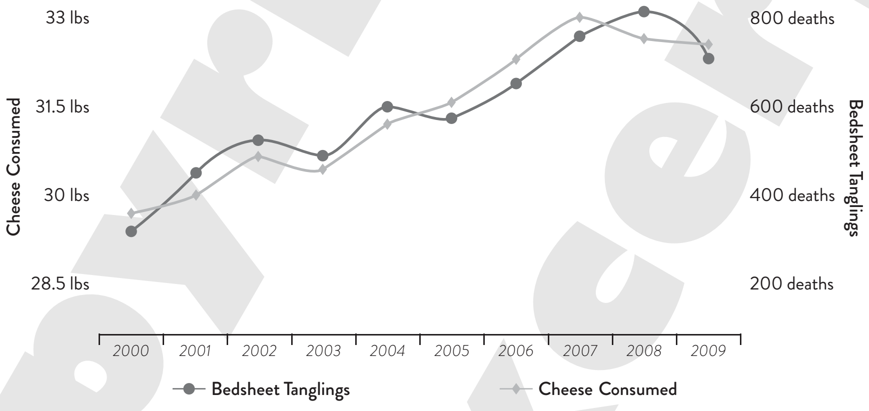


Figure 12.1: Spurious Correlation: Per Capita Cheese Consumption and Strangulation by Bedsheets

[1] http://www.tylervigen.com/spurious-correlations.

because our datasets are related and so often make sense. This is why it is so important to remember that correlation is only the exploratory stage: we can report correlations, and then we move on to more complex analyses.

There are several examples of correlations that are reported in our research and in this book, because we know the importance and value of understanding how things in our environment interrelate. In all cases, we reported Pearson correlations,[2] which is the correlation type most often used in business contexts today.

INFERENTIAL PREDICTIVE ANALYSIS

The third level of analysis, inferential, is one of the most common types conducted in business and technology research today. It is also called inferential predictive, and it helps us understand impacts of HR policies, organizational behavior and motivation, and how technology impacts outcomes like user satisfaction, team efficiency, and organizational performance. Inferential design is used when purely experimental design is not possible and field experiments are preferred—for example, in business, when data collection happens in complex organizations, not in sterile lab environments, and companies won't sacrifice profits to fit into control groups defined by the research team.

To avoid problems with "fishing for data" and finding spurious correlations, hypotheses are theory driven. This type of analysis is the first step in the scientific method. Many of us are familiar

2 Pearson correlations measure the strength of a linear relationship between two variables, called Pearson's r. It is often referred to as just correlation and takes a value between –1 and 1. If two variables have a perfect linear correlation, that is they move together exactly, $r = 1$. If they move in exactly opposite directions, $r = -1$. If they are not correlated at all, $r = 0$.

with the scientific method: state a hypothesis and then test it. In this level of analysis, the hypothesis must be based on a well-developed and well-supported theory.

Whenever we talk about *impacting* or *driving* results in this book, our research design utilized this third type of analysis. While some suggest that using a theory-based design opens us up to confirmation bias, this is how science is done. Well, wait—almost. Science isn't done by simply confirming what the research team is looking for. Science *is* done by stating hypotheses, designing research to test those hypotheses, collecting data, and then testing the stated hypotheses. The more evidence we find to support a hypothesis, the more confidence we have for it. This process also helps to avoid the dangers that come from fishing for data—finding the spurious correlations that might randomly exist but have no real reason or explanation beyond chance.

Examples of hypotheses tested with inferential analysis in our project include continuous delivery and architecture practices driving software delivery performance, software delivery positively affecting organizational performance, and organizational culture having a positive impact on both software delivery and organizational performance. In these cases, the statistical methods used were either multiple linear regression or partial least squares regression. These methods are described in more detail in Appendix C.

PREDICTIVE, CAUSAL, AND MECHANISTIC ANALYSIS

The final levels of analysis were not included in our research, because we did not have the data necessary for this kind of work. We will briefly summarize them here for the sake of completeness and to appease your curiosity.

- Predictive analysis is used to predict, or forecast, future events based on previous events. Common examples include cost or utilities predictions in business. Prediction is very hard, particularly as you try to look farther away into the future. This analysis generally requires historical data.
- Causal analysis is considered the gold standard, but is more difficult than predictive analysis and is the most difficult analysis to conduct for most business and technology situations. This type of analysis generally requires randomized studies. A common type of casual analysis done in business is A/B testing in prototyping or websites, when randomized data can be collected and analyzed.
- Mechanistic analysis requires the most effort of all methods and is rarely seen in business. In this analysis, practitioners calculate the exact changes to make to variables to cause *exact* behaviors that will be observed under certain conditions. This is seen most often in the physical sciences or in engineering, and is not suitable for complex systems.

CLASSIFICATION ANALYSIS

Another type of analysis is classification, or clustering, analysis. Depending on the context, research design, and the analysis methods used, classification may be considered an exploratory, predictive, or even causal analysis. We use classification in this book when we talk about our high-, medium-, and low-performance software delivery teams. This may be familiar to you in other contexts when you hear about customer profiles or market basket analysis. At a high level, the process works like this: classification variables are entered into the clustering algorithm and significant groups are identified.

In our research, we applied this statistical method using the tempo and stability variables to help us understand and identify if there were differences in how teams were developing and delivering software, and what those differences looked like. Here is what we did: we put our four technology performance variables—deployment frequency, lead time for changes, mean time to repair, and change fail rate—into the clustering algorithm, and looked to see what groups emerged. We see distinct, statistically significant differences, where high performers do significantly better on all four measures, low performers perform significantly worse on all four measures, and medium performers are significantly better than low performers but significantly worse than high performers. For more detail, see Chapter 2.

What Is Clustering?

For those armchair (or professional) statisticians who are interested, we used hierarchical clustering. We chose this over k-means clustering for a few reasons. First, we didn't have any theoretical or other ideas about how many groups to expect prior to the analysis. Second, hierarchical clustering allowed us to investigate parent-child relationships in the emerging clusters, giving us greater interpretability. Finally, we didn't have a huge dataset, so computational power and speed wasn't a concern.

THE RESEARCH IN THIS BOOK

The research presented in this book covers a four-year time period, and was conducted by the authors. Because it is primary research, it is uniquely suited to address the research questions we had in mind—specifically, what capabilities drive software delivery performance and organizational performance? This project was based on quantitative survey data, allowing us to do statistical analyses to test our hypotheses and uncover insights into the factors that drive software delivery performance.

In the next chapters, we discuss the steps we took to ensure the data we collected from our surveys was good and reliable. Then, we look into why surveys may be a preferred source of data for measurement—both in a research project like ours and in your own systems.

PART THREE TRANSFORMATION

We've presented our findings on which capabilities are important in producing better software delivery and organizational outcomes. However, taking this information and applying it to change your organization is a complex and daunting task. That's why we're delighted that Steve Bell and Karen Whitley Bell agreed to write a chapter on leadership and organizational transformation, sharing their experience and insights to guide readers in their own journey.

Steve and Karen are pioneers of Lean IT, applying principles and practices through a method-agnostic approach, drawing on a variety of practices—DevOps, Agile, Scrum, kanban, Lean startup, Kata, Obeya, strategy deployment, and others—as appropriate to the culture and situation, to coach and support leaders to develop high-performance practices and organizational learning capabilities.

In Part III, they draw on their experiences at ING Netherlands, a global bank with over 34.4 million customers worldwide and with 52,000 employees, including more than

9,000 engineers, to show the why and how of leadership, management, and team practices that enable culture change. This, in turn, enables sustainable high performance in a complex and dynamic environment.

Steve and Karen extend our view beyond the interrelationships of team, management, and leadership practices, beyond the skillful adoption of DevOps, and beyond the breaking down of silos—all necessary, but not sufficient. Here we see the evolution of holistic, end-to-end organizational transformation, fully engaged and fully aligned to enterprise purpose.

HIGH-PERFORMANCE LEADERSHIP AND MANAGEMENT

By Steve Bell and Karen Whitley Bell

"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 of these," the research shows, "have a measurable impact on an organization's profitability, productivity, and market share. These also have an impact on customer satisfaction, efficiency, and the ability to achieve organizational goals."[1] Yet, Nicole, Jez, and Gene also observe that "the role of leadership on technology transformation has been one of the more overlooked topics in DevOps."

Why is that? Why have technology practitioners continuously sought to improve the approach to software development and deployment as well as the stability and security of infrastructure and platforms, yet, in large part, have overlooked (or are unclear about) the way to lead, manage, and sustain these endeavors? This holds

[1] See Chapter 11, pp. 115–116.

for large legacy enterprises as well as digital natives. Let's consider this question not in the context of the past—why we haven't—but instead for the present and future: why we must improve the way we lead and manage IT[2] and, indeed, reimagine the way everyone across the enterprise views and engages with technology.

We are in the midst of a complete transformation in the way value is created, delivered, and consumed. Our ability to rapidly and effectively envision, develop, and deliver technology-related value to enhance the customer experience is becoming a key competitive differentiator. But peak technical performance is only one part of competitive advantage—necessary but not sufficient. We may become great at rapidly developing and delivering reliable, secure, technology-enabled experiences, but how do we know which experiences our customers value? How do we prioritize what we create so that each team's efforts advance the larger enterprise strategy? How do we learn from our customers, from our actions, and from each other? And as we learn, how do we share that learning across the enterprise and leverage that learning to continuously adapt and innovate?

The other necessary component to sustaining competitive advantage is a lightweight, high-performance management framework that connects enterprise strategy with action, streamlines the flow of ideas to value, facilitates rapid feedback and learning, and capitalizes on and connects the creative capabilities of every individual throughout the enterprise to create optimal customer experiences. What does such a framework look like—not in theory but in practice? And how do we go about improving and transforming our own leadership, management, and team practices and behaviors to become the enterprise we aspire to be?

[2] Note from Nicole, Jez, and Gene. The term "IT" is used throughout this chapter to refer to the software and technology process—much more than just a single function within the technology group at a company, like IT support or the helpdesk.

A HIGH-PERFORMING MANAGEMENT FRAMEWORK IN PRACTICE

Throughout this book, Nicole, Jez, and Gene discuss several Lean management practices that have been found to correlate with high organizational performance—specifically, "profitability, market share, and productivity . . . [in addition to measures that capture] broader organizational goals—that is, goals that go beyond simple profit and revenue measures."[3] Each of these practices is, in some way, synergistic and interdependent with the others. To illustrate how these leadership, management, and team practices work together, and to show the foundational thinking that enables them, we share the experiences of ING Netherlands, a global financial institution that pioneered digital banking and is recognized for its customer-centric technology leadership. Today, IT is leading ING's digital transformation effort.

"You have to understand why, not just copy the behaviors,"[4] says Jannes Smit, IT Manager of Internet Banking and Omnichannel at ING Netherlands, who, seven years ago, decided to experiment with ways to develop organizational learning among his teams. There are many ways we could describe this management practice in action. Perhaps the best way is to take you on a virtual visit—albeit from the pages of a book. (ING is happy to share the story of their learning, but they're not willing to show you what's on the walls!) We'll share with you the sights and sounds and experiences of a day at ING, showing you how practices, rhythms, and routines connect to create a learning organization and deliver high performance and value.

[3] See Chapter 2, p. 24.
[4] This and all other direct quotes from ING staff are personal communications with the authors of this chapter.

What you see today bears little resemblance to what we first observed as we periodically visited to facilitate what they called "boot camps" to rethink how Jannes and his managers led and managed teams. Like many enterprise IT organizations, they were located offsite from the main campus and were viewed by many as a function rather than as a vital contributor in realizing enterprise strategy. Today, we enter at the main corporate headquarters, where Jannes' teams are now located one floor below the C-suite. The space is open and light. After security, we pass through a large, open social area—coffee bars and snack kiosks overlooking gardens—designed to create intimate spaces to gather, visit, and share ideas. We then enter the Tribe's suite. Immediately to our left is a large room with glass walls, creating visibility to the space within. This is the Obeya room where the Tribe lead's work, priorities, and action items are visualized for the teams and anyone else who may schedule a meeting in this space or visit between meetings to update or review status. Here Jannes meets on a regular cadence with his direct reports, where they can quickly see and understand the status of each of his strategic objectives. Four distinct zones are visualized: strategic improvement, performance monitoring, portfolio roadmap, and leadership actions, each with current information about targets, gaps, progress, and problems. Color coding is used—red and green—to make problems immediately visible. Each IT objective ties directly, in measurable ways, to enterprise strategy (see Figure 16.1).

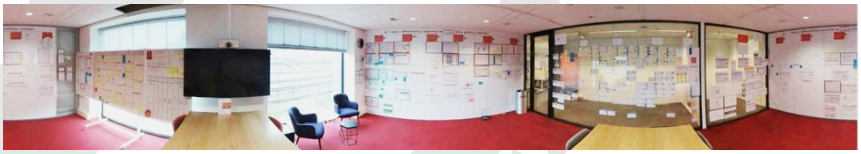
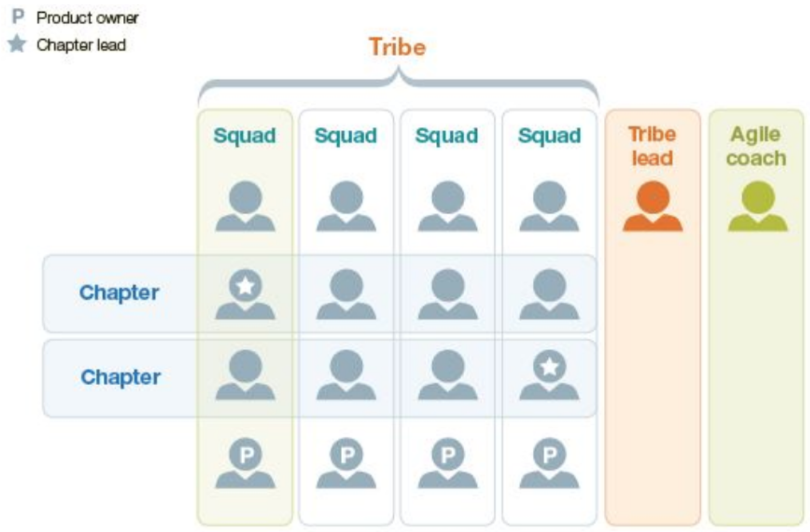


Figure 16.1: Leadership Obeya (360-Degree Panorama)

Two years ago, ING underwent a significant shift to a multi-dimensional, matrixed structure organized along lines of business, enabling the continuous flow of customer value (what Lean practitioners call value streams). Each line of business is organized as a tribe delivering a portfolio of related products and services (for example, the Mortgage Services Tribe). Each tribe is comprised of multiple self-steering teams, called squads, each responsible for a distinct customer mission (for example, the Mortgage Application Squad). Each squad is guided by a product owner, led (in case of IT) by an IT-area lead, and sized according to Bezos' Two Pizza Rule—no team can be so large that it would require more than two pizzas to feed them. Most squads are cross-functional, consisting of engineers and marketers, collaborating as a single team with a shared understanding of customer value. At ING, this team composition is referred to as BizDevOps. Recently, they identified a need for a new bridging structure which they plan to call a product area lead, to align multiple, closely related squads. This new role wasn't planned—it emerged through experience and learning. There are also chapters, comprised of members of the same discipline (for example, the Data Analytics Chapter), who are matrixed across squads and bring specialized knowledge to promote learning and advancement among squad members. And finally, there are centers of expertise, bringing together individuals with particular capabilities (for example, communications or enterprise architects—see Figure 16.2).

We move on from Jannes' Obeya, accompanied by Jannes' internal continuous improvement coaches: David Bogaerts, Jael Schuyer, Paul Wolhoff, Liedewij van der Scheer, and Ingeborg Ten Berge. Together, they form a small but effective Lean Leadership Expertise Squad and coach the leaders, chapter leads, product



Tribe

(collection of squads with interconnected missions)

- includes on average 150 people
- empowers tribe lead to establish priorities, allocate budgets, and form interface with other tribes to ensure knowledge/insights are shared

Agile coach

- coaches individuals and squads to create high-performing teams

Squad

(basis of new agile organization)

- includes no more than 9 people; is self-steering and autonomous
- comprises representatives of different functions working in single location
- has end-to-end responsibility for achieving client-related objective
- can change functional composition as mission evolves
- is dismantled as soon as mission is executed

Product owner

(squad member, not its leader)

- is responsible for coordinating squad activities
- manages backlog, to-do lists, and priority setting

Chapter

(develops expertise and knowledge across squads)

Chapter lead

- is responsible for one chapter
- represents hierarchy for squad members (re: personal development, coaching, staffing, and performance management)

Figure 16.2: ING's New Agile Organizational Model Has No Fixed Structure—It Constantly Evolves. (Source ING)

owners, and IT-area leads who, in turn, coach their chapter or squad members, creating a leveraged effect to change behavior and culture at scale.

Just ahead is a squad workspace—an open area with windows and walls that are covered in visuals (their own Obeya) that enable the squad to monitor performance in real time, and see obstacles, status of improvements, and other information of value to the squad. Across the middle of the space flows a row of adjustable-height tables, with adjustable-height chairs, enabling squad members to sit or stand, facing each other across their screens. The chairs are of different shapes and colors, making the space visually interesting and ergonomically sound. Squad visuals share some characteristics; the similarities in Obeya design enable colleagues outside the squad to immediately understand, at a glance, certain aspects of the work, promoting shared learning. Standard guidelines include visualizing goals, present performance and gaps, new and escalated problems, demand, WIP, and done work. Visualizing demand helps prioritize and keep the WIP load small. The visuals also have some differences, recognizing that the work of each squad is somewhat unique and each squad is the best judge of what information—and what visualization of that information—best serves them to excel at their work.

As we pass through, the squad is conducting its daily stand-up, where rapid learning and feedback takes place. Standing in front of a visual board displaying demand and WIP, each member briefly reports what she/he is working on (WIP), any obstacles, and what has been completed. As they speak, the visual is updated. These stand-ups usually last around 15 minutes; they have significantly reduced the time people spend in meetings compared to the meeting times before daily stand-ups became a way of work.

During the stand-ups, problems are not solved, but there is a routine in place to ensure they are rapidly resolved. If the problem requires collaboration with another squad member, it is noted, and those members will discuss it later in the day. If the problem requires IT-area lead support to resolve, the problem is noted and escalated. The IT-area lead may resolve it quickly, or take it to her/his stand-up to raise it with other IT-area leads or tribe leads to resolve. Once resolved, that information is rapidly relayed back through the channel. The problem remains visualized until it is resolved. Similarly, if the problem is technical in nature, it will be shared with the appropriate chapter or center of expertise. This pattern of vertical and horizontal communication is a leadership standard work practice called "catchball" (see Figure 16.3).

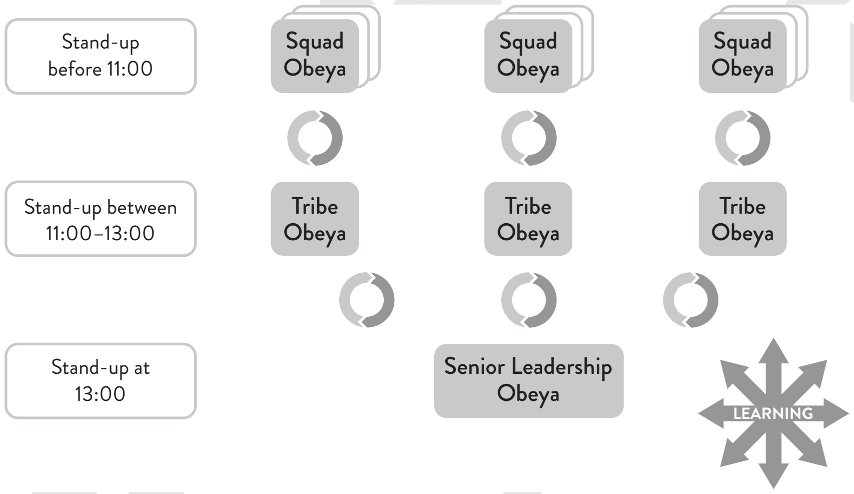


Figure 16.3: Stand-up and Catchball Rhythm

Using the same communication framework, other relevant learning is also relayed among squads, chapters, centers of expertise, and tribes, creating a natural vertical and horizontal flow of learning across all dimensions of the organization. This enables the squads to self-determine how best to craft their work to support overall enterprise strategy and enables effective prioritization. The tribe lead, in this case Jannes, also learns from the squad and chapter members, including lessons learned in their direct interaction with customers. This enables him to adapt his strategic thinking and goals and share insights with his peers and superiors.

This practice of rapid exchange of learning, enabling the frontline teams to learn about strategic priorities and the leaders to learn about customer experience from frontline team customer interaction, is a form of strategy deployment (Lean practitioners use the term Hoshin Kanri). It creates, at all levels, a continuous, rapid feedback cycle of learning, testing, validating, and adjusting, also known as PDCA.

In addition to regular stand-ups with squads, product owners, IT-area leads, and chapter leads, the tribe lead also regularly visits the squads to ask questions—not the traditional questions like "Why isn't this getting done?" but, rather, "Help me better understand the problems you're encountering," "Help me see what you're learning," and "What can I do to better support you and the team?" This kind of coaching behavior does not come easily to some leaders and managers. It takes real effort, with coaching, mentoring, and modeling (mentoring is being piloted within the Omnichannel Tribe, with plans for expansion) to change behavior from the traditional command-and-control to leaders-as-coaches where everyone's job is to (1) do the work, (2) improve the work, and (3) develop the people. The third objective—develop the people—is especially important in a technology domain, where automation is disrupting many technology jobs. For people to bring their best to the work that may, in fact, eliminate their current job, they need complete faith that their leaders value them—not just for their present work but for their ability to improve and innovate in their work. The work itself will constantly change; the organization that leads is the one with the people with consistent behavior to rapidly learn and adapt.

Not far from that squad space in a glass-enclosed meeting space with whiteboard-covered walls, a telepresence monitor, easel pads, and colorful, comfy chairs, we visit with Jordi de Vos, a young engineer whose entire career has been under Jannes' new way-of-working. Jordi is a chapter lead who also leads the effort toward one of the way-of-work strategic improvement objectives (recall that there are strategic improvement, performance monitoring, and portfolio roadmap strategic objectives). Jordi shares with others what he's learning about team security—the psychological safety for individuals to openly discuss problems and obstacles with no fear of harm or reprisal. He talks about this and other research he's discovering, how he's experimenting to learn what will resonate most among the squads, and what measurable changes are created and sustained. A fixed percentage of each squad's and chapter's time is allocated for improvement. Jordi says that the squads think of improvement activities as just regular work.

We ask Jordi what it's like to work within this culture. He reflects for a moment then shares a story. Jannes' tribes had been challenged by senior leadership to be twice as effective. "There was a tough deadline and lots of pressure. Our tribe lead, Jannes, went to the squads and said, 'If the quality isn't there, don't release. I'll cover your back.' So, we felt we owned quality. That helped us to do the right things."

Too often, quality is overshadowed by the pressure for speed. A courageous and supportive leader is crucial to help teams "slow down to speed up," providing them with the permission and safety to put quality first (fit for use and purpose) which, in the long run, improves speed, consistency, and capacity while reducing cost, delays, and rework. Best of all, this improves customer satisfaction and trust.

After this visit, we walk past more squad workspaces and more glass-enclosed meeting spaces, each with the same elements but different in their colors, textures, and furnishings. Back in the Leadership Obeya, we meet up with the coaching team for a healthy lunch and reflect on the many positive changes we've seen since our last visit. They share reflections on their current challenges and some of the approaches they are experimenting with to continue to spread and grow a generative culture, focusing on "going deep before going wide." Nevertheless, the pressure is there to scale wide and fast. Right now, one of the coaching team members is focusing on supporting culture change in just a few countries outside the Netherlands. Given that ING operates in over 40 countries, the discipline to allow time and attention for learning, rather than go for large scale change, is remarkable.

Another challenge the coaches are experimenting with is dispersed teams. With recent restructuring, some squads now have members from more than one country, so the coaching team is experimenting with, and measuring, ways to maintain the same high level of collaboration and learning among cross-border squads (it's very hard to virtually share two pizzas).

Not surprisingly, several of the most senior leaders and several other tribe lead peers want their own Obeya. The coaching team is hoping to approach this slowly enough so that real learning can occur. Transformational, generative leadership extends well beyond what is on the Obeya walls and the rhythm and routine of how you talk about it. "As a leader, you have to look at your own behaviors before you ask others to change," says Jannes. He will be the first to tell you that he is still learning. And in that, we believe, lies the secret to his success.

After lunch we head to the C-suite where we see a few of the senior leaders' Obeyas beginning to take shape. We run into Danny Wijnand, a chief design engineer who worked under Jannes until he was promoted last year to lead his own tribe. Danny reflects on the spread of this new way of work, beyond Jannes' tribes and out into the C-suite and across the rest of ING. "You get impatient wanting to speed their learning but then you realize you went through this yourself, and it took time. Storytelling is important, but they have to have their own learning."

Back again on the tribe floor, we visit with Jan Rijkhoff, a chapter lead. We wanted to learn about his chapter's current approach to problem solving. Over the years, they have experimented with different problem-solving methods, including A3, Kata, Lean startup, and others, and finally settled on a blend of elements that they found helpful, creating their own approach. In our walk today, we have seen evidence of multiple problem-solving initiatives in flight and visualized on the walls.

Their approach is to gather the right people who have experience and insights into the problem to rigorously examine the current condition. This rigor pays off, as the team gains insights that increase the probability of identifying the root cause rather than just the symptoms. With this learning, they form a hypothesis about an approach to improvement, including how and what to measure to learn if the experiment produces the desired outcomes. If the experiment is a success, they make it part of the standard work, share the learning, and continue to monitor to ensure the improvement is sustained. They apply this problem-solving approach at all levels of the organization. Sometimes a problem at a senior-leader level is analyzed and broken down into smaller parts, cascading to the chapter or squad level, for front-line analysis and controlled experimentation, with the learning feeding back up. “This approach works,” Paul tells us when we meet up again, “because it helps people to embrace change, letting people come up with their own ideas, which they can then test out.”

Amidst this colorful, creative work environment, with a philosophy of “make it your own,” the idea of standard work may seem to be antithetical, even counterproductive. After all, this is knowledge work. Consider the notion of process (the way something is done) and practice (doing something that requires knowledge and judgment). For example, Scrum rituals are process; the act of understanding customer needs and writing the code is practice. So, when teams have a standard way of work, whether that work is to release effective code or to conduct a team stand-up meeting, following that standard saves a lot of time and energy. At ING, standard work is established not by imitating a way of work that is prescribed in a book or used successfully by another company. Instead, a team within ING experiments with different approaches and agrees upon the one best way to do the work. That rhythm and routine is spread to all similar teams. As conditions change, the standard is reevaluated and improved.

We catch up with Jannes as he concludes his day with a visit to the Leadership Obeya—to add a few Post-It note updates and to see what updates have been made by others. We ask about his thoughts on the journey they’ve been on. “The beginning insight was that our teams were not learning and not improving,” he shared. “We were not able to get them to a level where they would be a continuously learning team. I saw that they wrestled with problems and other teams had solutions, and we were not able to bring them together to learn. When we were not able to learn as management, we were not able to help the teams to learn. We had to learn ourselves to become a learning team. We [his management team] experienced our own learning, then we went to the teams to help them learn to become a learning team."

We then asked about his approach to culture change. "Before, I never discussed culture," he said. "It was a difficult topic and I did not know how to change it in a sustainable way. But I learned that when you change the way you work, you change the routines, you create a different culture."

"Senior management is very happy with us," he adds with a broad smile, obviously proud of the people in his tribes. "We give them speed with quality. Sometimes, we may take a little longer than some of the others to reach green, but once we achieve it, we tend to stay green, when a lot of the others go back to red."

TRANSFORMING YOUR LEADERSHIP, MANAGEMENT, AND TEAM PRACTICES

We are often asked by enterprise leaders: How do we change our culture?

We believe the better questions to ask are: How do we learn how to learn? How do *I* learn? How can I make it safe for others to learn? How can I learn from and with them? How do we, together, establish new behaviors and new ways of thinking that build new habits, that cultivate our new culture? And where do we start?

At ING Netherlands, they began with a leader who asked himself these questions. He then brought on good coaches, tasked with challenging every person (including himself) to question assumptions and try new behaviors. He gathered his management team, saying, “Let’s try this together. Even if it doesn’t work, we will learn something that will help us to be better. Will you join me in this and see what we can learn?”

Each quarter his management team would come together for new learning and, over the next months, put that learning into practice. What, at first, felt uncomfortable for everyone became a little easier and, finally, became a habit—something they just did, just in time for the next learning cycle. They stretched and, just when they felt comfortable, stretched again. All along, they would reflect together and adjust when needed.

We recall in one boot camp session early on we challenged the management team members to develop simple leader standard work routines: visual management, regular stand-ups, and consistent coaching for their team members—replacing the long meetings and fire-fighting behaviors they were accustomed to. To develop this new way of working, first they needed to understand how they currently spent their time. The skepticism and discomfort were obvious; nevertheless, for several weeks each of them recorded and measured how they spent their time each day. They shared what they learned with each other, and together developed new ways to work.

When we returned for the next boot camp three months later, Mark Nijssen, one of the managers, welcomed us by saying, “I’ll never go back to the old way of working again!” Not only was adoption of basic leader standard work successful in helping them improve their effectiveness, they also managed to achieve the goal of making 10% of their time available to work on what they choose.

This willingness to experiment with new ways of thinking and working has led ING to where they are today. But it’s important to recognize that there is no checklist or playbook. You can’t "implement" culture change. Implementation thinking (attempting to mimic another company's specific behavior and practices) is, by its very nature, counter to the essence of generative culture.

At the end of this chapter is a table representing many of the practices described in this virtual visit to ING. Those marked with an (*) are practices that research shows to correlate with high performance. It's our hope that future research will explore the full range of practices listed here. This table is not to be used as a checklist but rather as a distillation or general guidelines for developing your own behaviors and practices (see Figure 16.4).

As you have seen in our virtual visit to ING, a high-performance culture is far more than just the application of tools, the adoption of a set of interrelated practices, copying the behaviors of other successful organizations, or the implementation of a prescribed, expert-designed framework. It is the development, through experimentation and learning guided by evidence, of a new way of working together that is situationally and culturally appropriate to each organization.

As you begin your own path to creating a learning organization, it's important to adopt and maintain the right mindset. Below are some suggestions we offer, based on our own experiences in helping enterprises evolve toward a high-performing, generative culture:

- Develop and maintain the right mindset. This is about learning and how to create an environment for shared organizational learning—not about just doing the practices, and certainly not about employing tools.
- Make it your own. This means three things:
  - Don't look to copy other enterprises on their methods and practices, or to implement an expert-designed model. Study and learn from them, but then experiment and adapt to what works for you and your culture.

	Team Practices	Management Practices	Leadership Practices
Culture	*Foster generative culture	*Foster generative culture	*Foster generative culture
	*Build quality in, continuously measure and monitor	*Focus on quality, protect teams to ensure quality	*Focus on quality, protect teams to ensure quality
	Focus on promoting organizational learning	Focus on promoting organizational learning	Focus on promoting organizational learning
		*Provide teams with time for improvement and innovation	*Provide teams with time for improvement and innovation
Organizational Structure			*Align, measure, and manage to flow (matrixed, cross-functional value stream organization structure)
		Establish small, cross-functional, multiskilled teams; support bridging structures so teams can easily communicate and collaborate	Enable and support cross-skilling to reduce expert-dependent bottlenecks, and form communities of expertise
			Establish and support internal coaches and the appropriate infrastructure to scale and sustain them
Direct Learning and Alignment to Value	*Engage, learn from, and validate with customers (Gemba)	*Engage with and learn from customers and teams (Gemba)	*Engage with and learn from customers, teams, supply chain partners, and other stakeholders (Gemba)
	*Understand & visualize customer value, identify measurable targets for quality	*Understand & visualize customer value, identify measurable targets for quality	
	*Practice creativity as part of overall work	*Practice creativity as part of overall work, encourage team members to utilize this time to learn and innovate	*Budget for and allocate time for creativity (i.e., Google's 20% target)
Strategy Deployment	*Visualize team goals and targets, understand how these targets advance enterprise strategy	Help teams to set and visualize goals and targets, understand and communicate how these targets advance enterprise strategy (catchball)	Practice strategy deployment, visualize all goals and near-term targets, communicate this clearly to managers and help them set appropriate targets and initiatives
	*Actively monitor and visualize performance to goals/targets	*Actively monitor and visualize performance to goals/targets	*Actively monitor and visualize performance to goals/targets
			Eliminate unnecessary controls, invest instead in process quality and team autonomy and capability (*teams that reported no approval process or used peer review achieved higher software delivery performance)
Improve Flow Through Analysis and Disciplined Problem Solving	Visualize & analyze workflow, identify obstacles to flow, (process/value stream mapping & analysis); *understand the connection between the work they do and its positive impact on customers	Visualize and analyze workflow, identify obstacles to flow, (process/value stream mapping & analysis), help teams understand how they support larger value stream	Visualize and analyze overall value stream flows (enterprise architecture), identify systemic obstacles to flow, prioritize and support mapping and analysis of lower-level supporting flows
	Prioritize obstacles to customer value and experience, and team targets and goals	Prioritize obstacles to customer value and experience, and team targets and goals	Prioritize systemic obstacles to flow
	Apply disciplined problem solving to prioritized problems, analyze to identify root causes	Apply disciplined problem solving to prioritized problems, analyze to identify root causes	Apply disciplined problem solving to complex systemic issues to identify strategic improvement themes and targets (strategy deployment), apply learning to update standard work
	Escalate cross-functional and systemic problems	Coordinate cross-functional problem solving, solve or escalate systemic problems	Cascade prioritized problem solving targets to the appropriate stakeholders through catchball PDCA
	Form hypotheses about root causes, design and conduct controlled experiments, measure results, communicate learnings, repeat if needed, incorporate improvements	Form hypotheses about root causes, design and conduct controlled experiments, measure results, communicate learnings, repeat if needed, incorporate improvements	Learn from organization-wide PDCA cycles, and repeat learning/improvement cycles
Way of Work, Rhythm, & Routine	*Visualize, measure, and monitor workflow, monitor for deviations, respond to deviations appropriately	*Visualize, measure, and monitor workflow, monitor for deviations, respond to deviations appropriately	*Visualize, measure, and monitor workflow, monitor for deviations, respond to deviations appropriately
	*Break demand into small elements (MVP's) and release regularly and often		
	*Visualize demand, WIP, and "done" (kanban)	*Visualize demand, WIP, and "done" (kanban)	*Visualize demand, WIP, and "done" (kanban)
	*Minimize and visualize WIP	*Minimize and visualize WIP	*Minimize and visualize WIP
	Prioritize demand to goals and targets	Prioritize demand to goals and targets	Prioritize demand to goals and targets
	Develop & practice team standard work (rhythm & routine)	Develop & practice leader standard work (rhythm & routine)	Develop & practice leader standard work (rhythm & routine)
	Conduct daily stand-ups with standard routine, escalate obstacles as needed (catchball)	Conduct daily stand-ups with team leads, standard routine, resolve or bridge/escalate obstacles as needed (catchball)	Conduct stand-ups with direct reports with standard routine on a regular cadence, resolve escalated obstacles (catchball)
	Support team and peer learning	Coach team members; support team learning	Coach managers, have your own coach
	Conduct regular cadence of retrospectives (work and way of work)	Conduct regular cadence of retrospectives (work and way of work)	Conduct regular cadence of retrospectives (work and way of work)

Figure 16.4: High-Performance Team, Management, and Leadership Behaviors and Practices
(not a complete list, for a larger, downloadable version visit https://bit.ly/high-perf-behaviors-practices)

  - Don't contract it out to a large consulting firm to expediently transform your organization or to implement new methodologies or practices for you. Your teams will feel that these methodologies (Lean, Agile, whatever) are being done *to* them. While your current processes may temporarily improve, your teams will not develop the confidence or capability to sustain, continue to improve, or to adapt and develop new processes and behaviors on their own.
  - Do develop your own coaches. Initially you may need to hire outside coaching to establish a solid foundation, but you must ultimately be the agent of your own change. Coaching depth is a key lever for sustaining and scaling.
- You, too, need to change your way of work. Whether you are a senior leader, manager, or team member, lead by example. A generative culture starts with demonstrating new behaviors, not delegating them.
- Practice discipline. It was not easy for Jannes' management team to record and reflect on how they spent their time or try new things they weren't initially comfortable with in front of the people who reported to them. Change takes discipline and courage.
- Practice patience. Your current way of work took decades to entrench. It's going to take time to change actions and thought patterns until they become new habits and, eventually, your new culture.
- Practice practice. You just have to try it: learn, succeed, fail, learn, adjust, repeat. Rhythm and routine, rhythm and routine, rhythm and routine . . .

As you learn a new way of leading and working, you, and those you bring along with you on this journey, will explore, stretch, make some mistakes, get a lot right, learn, grow, and keep on learning. You'll discover better and faster ways to engage, learn, and adapt to changing conditions. In doing so, you'll improve quality and speed in everything you do. You'll grow your own leaders, innovate, and outperform your competition. You'll more rapidly and effectively improve value for customers and the enterprise. As the research shows, you'll "have a measurable impact on an organization's profitability, productivity, and market share. These also have an impact on customer satisfaction, efficiency, and the ability to achieve organizational goals."

We wish you all the best on your learning journey!

Steve and Karen