The DevOps Phenomenon

AN EXECUTIVE CRASH COURSE

ANNA WIEDEMANN, NICOLE FORSGREN, MANUEL WIESCHE, HEIKO GEWALD, AND HELMUT KRCMAR

Research for Practice combines the resources of the ACM Digital Library, the largest collection of computer science research in the world, with the expertise of the ACM membership. In every RfP column experts share a short curated selection of papers on a concentrated, practically oriented topic.

A traditional software company releases its flagship product infrequently—as rarely as every few years. Each release can include hundreds of new features and improvements. Because releases are infrequent, users can grow impatient waiting for each new release and are thankful when it finally arrives.

Disappointment sets in, however, when bugs are found and features don't work as expected. Under great stress and with great turmoil, an emergency release is produced and put into production (hurried through the regular release process, often achieved by skipping tests), which has still more bugs, and the process repeats with more emergency releases, leading to more frustration, stress, and disappointment. Worse yet, new business opportunities are missed or ignored because of doubt, uncertainty, and distrust in the IT department's ability to deliver value.

Isn't there a better way?

Such practices are a thing of the past for companies that subscribe to the DevOps method of software development and delivery. New releases are frequent: often weekly or daily. Bugs are fixed rapidly. New business

opportunities are sought with gusto and confidence. New features are released, revised, and improved with rapid iterations. In one case study, a company was able to provide a new software feature every 11 seconds.[17]

Which of these software teams would you rather be? Which of these companies will win during their industry's digital transformation?

DevOps presents a strategic advantage for organizations when compared with traditional software-development methods (often called *phase-gate* or *waterfall*.[7] Leadership plays an important role during that transformation. In fact, Gartner predicts that CIOs who haven't transformed their teams' capabilities by 2020 will be displaced.[9]

PROMISES AND CHALLENGES FOR DIGITAL TRANSFORMATION

For organizations hoping to capture market share and deliver value faster (or even just deliver software more safely and securely), DevOps promises both speed and stability.[4] It has developed as a prominent phenomenon of digital transformation in modern organizations that use software to deliver value to their customers in industries including banking, retail, and even manufacturing. DevOps combines activities of software development and delivery to enhance the speed of getting new software features to customers. This leads to higher customer satisfaction and profitability, which are important outcomes at the organization level. It also leads to important team-level outcomes such as better collaboration among different departments (such as developers, testers, and IT

operations) and improved work-life balance.

Executing a successful DevOps transformation isn't without its challenges. Organizations and software products vary in maturity and implementation, making transformation efforts difficult to design and deploy across teams and organizations. Most importantly, for DevOps to truly deliver value, it must include more than just tooling and automation—so simply purchasing and installing a solution isn't sufficient. As outlined here, DevOps includes culture, process, and technology. Indeed, success stories abound: Companies such as Kaiser Permanente, Capital One, Target, Starbucks, and ING have adopted DevOps methods, allowing them to deliver software for key applications in just seconds.

DevOps enhances automation from applications to infrastructure provisioning. Continuous delivery supports automation and enables faster time to market and agile software development with fast feedback cycles. As the phenomenon is relatively new in practice, practitioners report on struggles with issues such as leadership and cultural transformation, implementing continuous delivery pipelines, and integrating a culture of collaboration in team settings.[8] Each of these areas would benefit from examination and guidance by formal research to test and augment the valuable experiences of those in industry.

A MOVE AWAY FROM TRADITIONAL PROJECT MANAGEMENT

DevOps methodology is marked by a change in how software delivery is treated: moving from a discrete project to an ongoing product. The traditional way of

delivering software (referred to previously as *phase-gate* or *waterfall*) happens with the help of project management. In this paradigm, the project typically "ends" with the first major release of the new software or a set of new features delivered in a major incremental release. In general, the project team dissolves following a new release, and the responsibility for running the system is then "thrown over the wall" to operations. The operations team takes over responsibility for further changes and incident management. This leads to several problems:

- Developers are not responsible for running the system they built and therefore do not understand if tradeoffs appear in creating and running the system, notably in the scalability and reliability of the software. This can lead to the same problems being perpetuated in future software releases, even if they are well understood by the operations team.
- The operations team is responsible for maintaining a highly reliable and stable system. Each new line of software deployment introduces change, and therefore leads to instability. This leads to a mismatch in incentives, where accepting new software from developers introduces risk (instability) and uncertainty (because less or no visibility into the development process gives them little insight into the software they are inheriting). Even if new components are of high quality, all new code adds complexity to the system and risk that the software was not developed with scalability and reliability in mind—key factors that operations must address and support in new software, as well as existing software.
- Stakeholders upstream (that is, earlier in the

development process) from the production environment, including developers and the business, are not able to receive any feedback about performance until the first complete release. This generally takes place several months after project approval. Furthermore, not all features in software deliver value, and speeding up feedback to the development team and business allows for faster iteration to refine the software. This leads to wasted time and resources on features that don't ultimately deliver value. For example, research from Microsoft shows that only one-third of well-designed features delivered value to the customer.[14]

In contrast, DevOps calls for a shift to product-based management. Practically, this means there is no more "end date" to projects, and teams instead deliver features—and therefore value—continuously. An important part of achieving this is integrating teams throughout the value stream, from development through operations; some organizations are even including business stakeholders. In this model, software is a product that is maintained as a product, with delivery and value metrics being tracked by the business continuously.

STATE-OF-THE-ART RESEARCH AND PRACTICE ON DEVOPS

One of the biggest challenges (and complaints!) in industry is the lack of a formal definition for DevOps. Many practitioners argue that this is intentional, because it allows teams and organizations to adopt a definition that works for them. In addition, they point out that having a formal definition of agile (coded in the Agile Manifesto) hasn't solved the problem of definition sprawl, and the

resulting confusion around what is truly meant when an organization says it is "going agile" still plagues the industry. This lack of understanding can be challenging, so we present some common definitions for reference here.

- "DevOps is a software development and delivery methodology that provides... increased speed and stability while delivering value to organizations."[4]
- "DevOps, whether in a situation that has operations engineers picking up development tasks or one where developers work in an operations realm, is an effort to move the two disciplines closer."[16]
- "DevOps, a compound of 'development' and 'operations,' is a software development and delivery approach designed for high velocity."[17]

And yet, these definitions focus on the outcome (value through speed and stability, via Forsgren[4]) or the foundations of the discipline (bringing together development and operations, via Roche[16] or Sebastian et al.[17] If one wants to understand the components of DevOps methods, perhaps the most common presentation of these is summarized as CALMS: culture, automation, lean, measurement, and sharing.[6,12] Here is a brief definition of each component:

- *Culture.* Integration of mutual trust, willingness to learn and continuous improvement, constant flow of information, open-mindedness to changes and experimentation between developers and operations.
- *Automation.* Implementing deployment pipelines with a high level of automation (most notably continuous integration/continuous delivery) and comprehensive test automation.

- *Lean.* Applying lean principles such as minimization of work in progress, as well as shortening and amplification of feedback loops to identify and minimize value flow breaks to increase efficiency.
- *Measurement.* Monitoring the key system metrics such as business or transactions metrics and other key performance indicators.
- *Sharing.* Sharing knowledge in the organization and across organizational boundaries. Team members should learn from each other's experiences and proactively communicate.

AN INTERPRETATION OF CALMS

This article uses the five core elements of CALMS as a framework.

Culture

Working as part of a DevOps team requires a culture of collaboration within a cross-functional team setting. In its ideal state, DevOps uses a so-called cross-functional team, which means that groups made up of developers, testers, quality assurance professionals, and IT operations engineers all work together to develop and deliver software. In this way, they are familiar with each others' work and challenges (a common phrase to describe this is "to have empathy"), which helps them create and maintain better software. For example, because they see the challenges in maintaining scalable and reliable infrastructure encountered by operations professionals, developers write more scalable and reliable code in collaboration with operations staff.

Automation

The next principle, automation, requires a suite of DevOps tools.[2,13] Following are a few examples of available automation tooling:

Build

- *Build.* Tools for the software development and service life cycle, including compiling code, managing dependencies, creating documentation, conducting tests, and deployments of an application in different stages.
- *Continuous integration.* Constant testing of new software components.

Release

- *Release automation.* Packaging and deploying a software component from development across all environments to production.
- *Version control.* Managing changes to the program and collecting other information. Version control is a component of configuration management.
- *Test automation.* Using software to execute tests and repeating activities.

Deployment

- *Configuration management.* Reducing production provisioning and configuration maintenance with the help of reproducing the production system on development stages.
- *Continuous delivery.* A combination of principles, practices, and patterns designed to make uninterrupted deployments.

Operations

- *Logging.* Traces are essential for application management; everything must be logged.
- *Monitoring.* Identification of infrastructure problems before the customers notice them. Monitoring storage, network traffic, memory consumption, etc.

Continuous integration and delivery reduce the cost and risk of releasing software. They do this by combining automation and good practices to consistently, reliably, and repeatably perform work (such as tests and builds) that enables fast feedback and builds quality in during the software-delivery process. This helps teams deliver features faster and more reliably and, in turn, achieve faster value delivery for the organization. The highest-performing teams are able to deploy 46 times more frequently with 2,555 times shorter lead times than low performers. Failure rates are seven times less, and they are able to recover 2,604 times faster than their lower-performing peers.[8]

Lean

"Building quality in"—referenced in the previous paragraph—is a key tenet in DevOps, and a principle also found in *lean*. Applied to DevOps, this means teams look for opportunities to remove waste, leverage feedback loops, and optimize automation.

Let's look at an example. DevOps teams differ in size and product responsibility. In some models, a single team conducts all software development and delivery activities, including development, testing, delivery, and maintenance.

They are ultimately responsible for the complete software-delivery life cycle of the software products (and may be responsible for more than one product), delivering value to the business. Lean processes allow for quick iterations and feedback throughout the development and delivery process to improve quality and build faster and more reliable systems. Examples include working in small batches to enable fast flow through the development pipeline, limiting work in process, fixing errors as they are discovered vs. at the end, and "shifting left" on security input.

These practices may sound familiar; similar practices have driven quality and value in manufacturing. (For a great story about lean manufacturing, check out episode 561 of This American Life,[11] which discusses NUMMI (New United Motor Manufacturing Incorporated), the joint venture between Toyota and GM in Fremont, CA.)

Measurement

Measurement is another core aspect of DevOps. The ability to monitor and observe systems is important, because software development and delivery are essentially dealing with an invisible inventory that interacts in complex ways that cannot be observed. (This is in contrast to traditional physical manufacturing systems such as an automobile assembly line, described in the NUMMI case.)

Through effective monitoring, teams are able to track, watch, measure, and debug their systems throughout the software-delivery life cycle. It should be noted that metrics are also a tool for quality assurance, and measurements from several sources should be leveraged.[9]

Sharing

Sharing of knowledge and information enables successful DevOps teams and helps amplify their success. By sharing practices—both successes and failures—within teams, across the organization, and across the industry, teams benefit from the learning of others and improve faster. While others have pointed out that sharing is possible in any domain and any methodology, DevOps has adopted this as a cultural norm, and many in the industry report that the field is much more collaborative than their prior work in tech.

Internal collaboration may include work shadowing or job swapping: developers are involved in operations and maintenance activities (for example, developers may even "take the pager"), and operations engineers rotate in to development and test roles, learning essential components of design and test work. In many cases, all cross-functional team members participate in the same meetings, which gives them shared context. Cross-industry sharing often takes places at conferences, with dozens of DevOpsDays and other community-organized events sprouting up around the world.

The application of these principles leads to better outcomes: for individuals (seen in reduced burnout and greater job satisfaction), for teams (seen in better software delivery outcomes and better team cultures), and for organizations (seen in improved performance in measures such as profitability, productivity, customer satisfaction, and efficiency.[7,21]

Although DevOps has been an important movement in industry for more than a decade, it hasn't received much

attention from the academic community until recently. And while CALMS principles are not always referred to using these terms, they do appear in existing research (e.g., Fitzgerald and Stol[3]).

RESEARCH SUMMARIES

The core insights of some prior DevOps-related research follow:

Journal articles

- Fitzgerald and Stol (2017). *The Journal of Systems and Software*[3]

Presents a continuous software-engineering pipeline and a research agenda for different continuous processes including DevOps and BizDev (business strategy and development).

- Forsgren, Sabherwal, and Durcikova (2018). *European Journal of Information Systems*[10]

Highlights the roles adopted when sharing and sourcing knowledge in a system management context, and identifies strategies for optimizing outcomes.

- Forsgren, Durcikova, Clay, and Wang (2016). *Communications of the AIS*[5]

Identifies the most important system and information characteristics of tools for users who maintain systems.

- Dennis, Samuel, and McNamara (2014). *Journal of Information Technology*[1]

Highlights that maintenance effort should be considered during the design phase and calls this DFM (design for maintenance). The authors present insights into how the links among documents should affect both the

maintenance effort and use.

- Sharma and Rai (2015). *European Journal of Information Systems*[19]

Investigates how an organization's computer-aided software engineering adoption decision is influenced by individual factors of IS leaders and technological factors.

- Shaft and Vessey (2006). *Journal of Management Information Systems*[18]

Aims to examine software maintenance as interlinking comprehension and modification; the relationship between these two factors is moderated by cognitive fit.

Conferences

- Trigg and Bødker (1994). ACM Conference on Computer Supported Cooperative Work[20]

Examines how people tailor their shared PC environment and presents an understanding of how software development tailoring can be helpful in designing systems that better fit the demands.

- Lwakatare et al. (2016). Hawaii International Conference on System Sciences[15]

Investigates key challenges of DevOps adoption in embedded system domains.

- Wiedemann and Wiesche (2018). European Conference on Information Systems[22]

Presents an ideal skill set that DevOps team members should adopt to manage the software delivery life cycle.

PRACTICAL CHALLENGES AND OPPORTUNITIES

Implementing DevOps presents several challenges. First, technology—and the resulting organizational

transformation—are difficult, but strong leadership can help. As many practitioners note, managing and enabling the cultural changes inside an organization can be a more difficult and important challenge than implementing the technical changes. Good leadership is vital. One approach studied in several contexts, including DevOps, is transformational leadership; this style uses five dimensions (vision, intellectual stimulation, inspirational communication, supportive leadership, and personal recognition) to inspire and guide teams.

Second, DevOps requires a custom solution for each organization. Each context is unique, and a prescriptive approach to DevOps implementation and adoption is unlikely to be successful. Teams and organizations pose a unique sets of challenges and cultural norms. Each one should adopt and adapt its own approach to achieve DevOps success. The adaptation of DevOps should include development of not only technical, but also cultural, process, and measurement practices. The work of creating a unique and seemingly ad hoc technology transformation journey is difficult and may be daunting, so many organizations look for step-by-step guides. However, these are not likely to provide solutions (beyond basic advice such as "automate your toolchain") and are usually offered by those trying to sell you something.

Third, each DevOps solution should encompass a holistic view, consisting of automation (including tools and architecture), process, and culture. In many traditional approaches, specialist knowledge in one area (e.g., development) is leveraged to accomplish a task before passing it off to another group. In DevOps, a move from

this high specialization to include a broad understanding of more areas is necessary. (Some call this *T-shaped knowledge*, with the top part of the "T" representing broad knowledge, while the stem of the T represents deep understanding in one area of expertise.)

This allows people to understand how their work will affect and interact with more areas of the technical stack; this often requires significant additional learning and responsibilities in the transition to DevOps. Organizations should provide training and education, and not just expect technologists to augment their learning independently. Note that while some technologists consider this expansion of responsibilities and knowledge exciting, others may push back, especially those who are just a few years from retirement and comfortable in their work roles, or who see sharing information about their roles as a risk to job security. Organizations will need to consider these training and cultural challenges in particular and respond accordingly. (As already noted, DevOps is about much more than just technology!)

CONCLUSION

DevOps is about providing guidelines for faster time to market of new software features and achieving a higher level of stability. Implementing cross-functional, product-oriented teams helps bridge the gaps between software development and operations. By ensuring their transformations include all of the principles outlined in CALMS (culture, automation, lean, metrics, and sharing), teams can achieve superior performance and deliver value to their organizations. DevOps is often challenging,

but stories from across the industry show that many organizations have already overcome the early hurdles and plan to continue their progress, citing the value to their organizations and the benefits to their engineers.

References

1. Dennis, A.R., Samuel, B.M., McNamara, K. 2014. Design for maintenance: how KMS document linking decisions affect maintenance effort and use. *Journal of Information Technology* 29(4), 312-326.
2. Ebert, C., Gallardo, G., Hernantes, J., Serrano, N. 2016. DevOps. *IEEE Software* 33(3), 94-100; https://ieeexplore.ieee.org/document/7458761.
3. Fitzgerald, B., Stol, K.-J. 2017. Continuous software engineering: a roadmap and agenda. *Journal of Systems and Software* 123, 176–189.
4. Forsgren, N. 2018. DevOps delivers. *Communications of the ACM* 61(4), 32-33; https://dl.acm.org/citation.cfm?id=3174799.
5. Forsgren, N., Durcikova, A., Clay, P. F., Wang, X. 2016. The integrated user satisfaction model: assessing information quality and system quality as second-order constructs in system administration. *Communications of the Association for Information Systems* 38, 803-839.
6. Forsgren, N., Humble, J. 2015. The role of continuous delivery in IT and organizational performance. In *Proceedings of the Western Decision Sciences Institute*.
7. Forsgren, N., Humble, J., Kim, G. 2018. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High-performing Technology Organizations*. Portland, Oregon: IT Revolution Press.

8. Forsgren, N., Humble, J., Kim, G. 2018. Accelerate: state of DevOps report: strategies for a new economy. DORA (DevOps Research and Assessment) and Google Cloud; https://cloudplatformonline.com/rs/248-TPC-286/images/DORA-State%20of%20DevOps.pdf.
9. Forsgren, N., Kersten, M. 2018. DevOps metrics. *Communications of the ACM* 61(4), 44-48; https://dl.acm.org/citation.cfm?id=3200906.3159169.
10. Forsgren, N., Sabherwal, R., Durcikova, A. 2018. Knowledge exchange roles and EKR performance impact: extending the theory of knowledge reuse. *European Journal of Information Systems* 27(1), 3-21.
11. Glass, I. Episode 561, "NUMMI 2015." *This American Life;* https://www.thisamericanlife.org/561/nummi-2015.
12. Humble, J., Molesky, J. 2011. Why enterprises must adopt DevOps to enable continuous delivery. Cutter IT Journal 24(8), 6-12; https://www.cutter.com/sites/default/files/itjournal/fulltext/2011/08/itj1108.pdf.
13. Humble, J. 2018. Continuous delivery sounds great, but will It work here? *Communications of the ACM* 61(4), 34-39; https://dl.acm.org/citation.cfm?id=3173553.
14. Kohavi, R., Crook, T., Longbotham, R., Frasca, B., Henne, R., Ferres, J. L., Melamed, T. 2009. Online experimentation at Microsoft, Data Mining Case Studies 11.
15. Lwakatare, L.E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H.H., Bosch, J., Oivo, M. 2016. Towards Devops in the embedded systems domain: Why is it so hard? In *Proceedings of the Hawaii International Conference on System Sciences*.
16. Roche, J. 2013. Adopting DevOps practices in quality

assurance. *Communications of the ACM* 56(11), 38-43; https://dl.acm.org/citation.cfm?id=2524721.
17. Sebastian, I. M., Ross, J. W., Beath, C., Mocker, M., Moloney, K. G., Fonstad, N. O. 2017. How big old companies navigate digital transformation. *MIS Quarterly Executive* 16(3), 197-213.
18. Shaft, T.M., Vessey, I. 2006. The role of cognitive fit in the relationship between software comprehension and modification. *MIS Quarterly* 30(1), 29-55.
19. Sharma, S., Rai, A. 2015. Adopting IS process innovations in organizations: the role of IS leaders' individual factors and technology perceptions in decision making. *European Journal of Information Systems* 24(1), 23-37.
20. Trigg, R.H., Bødker, S. 1994. From implementation to design: tailoring and the emergence of systematization. In *CSCW '94: Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 45-54; https://dl.acm.org/citation.cfm?id=192869.
21. Wiedemann, A. 2017. A new form of collaboration in IT teams—exploring the DevOps phenomenon. In *Proceedings of the Pacific Asia Conference on Information Systems*, 1-12.
22. Wiedemann, A., Wiesche, M. 2018. Are you ready for Devops? Required skill set for Devops teams. In *Proceedings of the European Conference on Information Systems*.

Acknowledgements

The authors would like to thank our anonymous reviewers, whose guidance and thoughtful feedback helped us to shape and refine this paper.

Anna Wiedemann *is a research assistant at Neu-Ulm University of Applied Sciences and a doctoral candidate at the Technical University of Munich. Her research interests include topics in the areas of DevOps, project management, and business-IT alignment. Her work has been presented at numerous conferences and published in the* International Journal of IT/Business Alignment and Governance.

Nicole Forsgren *does research and strategy at Google Cloud. She is co-author of* Accelerate: The Science of Lean Software and DevOps *and is best known as lead investigator on the largest DevOps studies to date. She has been a successful entrepreneur (with an exit to Google), professor, performance engineer, and sysadmin. Her work has been published in several peer-reviewed journals.*

Manuel Wiesche *is a postdoctoral researcher at the Chair for Information Systems, Department of Informatics at the Technical University of Munich (TUM). His current research experiences and interests include project management, platform ecosystems, and IT services. His research has been published at a number of refereed conference proceedings, the MIS Quarterly and other numerous peer-reviewed journals.*

Heiko Gewald *is research professor of information management at Neu-Ulm University of Applied Sciences and director of the Center for Research on Service Sciences. His research focuses on IT management, health IT, and the use of digital resources by the aging generation. He is a frequent speaker at conferences on these matters, and his work has been published in numerous peer-reviewed journals.*

Helmut Krcmar *is a full professor of information systems, Department of Informatics at the Technical University of Munich and speaker of the directorate of fortiss GmbH, the Research Institute of the Free State of Bavaria for Software and Systems. His research interests include digital transformation, information and knowledge management, and platform-based ecosystems management of IT-based service systems. His work has been published widely in peer-reviewed journals. He is a Fellow of the Association for Information Systems.*

CONTENTS