Azure Annual DevOps Report

Microsoft

sogeti
Part of Capgemini

Enterprise DevOps Report
2020-21

Table of Contents

Executive Summary 3

Introduction to Enterprise DevOps 6

The Developer Velocity Index (DVI) 9

Product Management 11

Distributed Teams and Remote Working 17

Governance 23

Quality 30

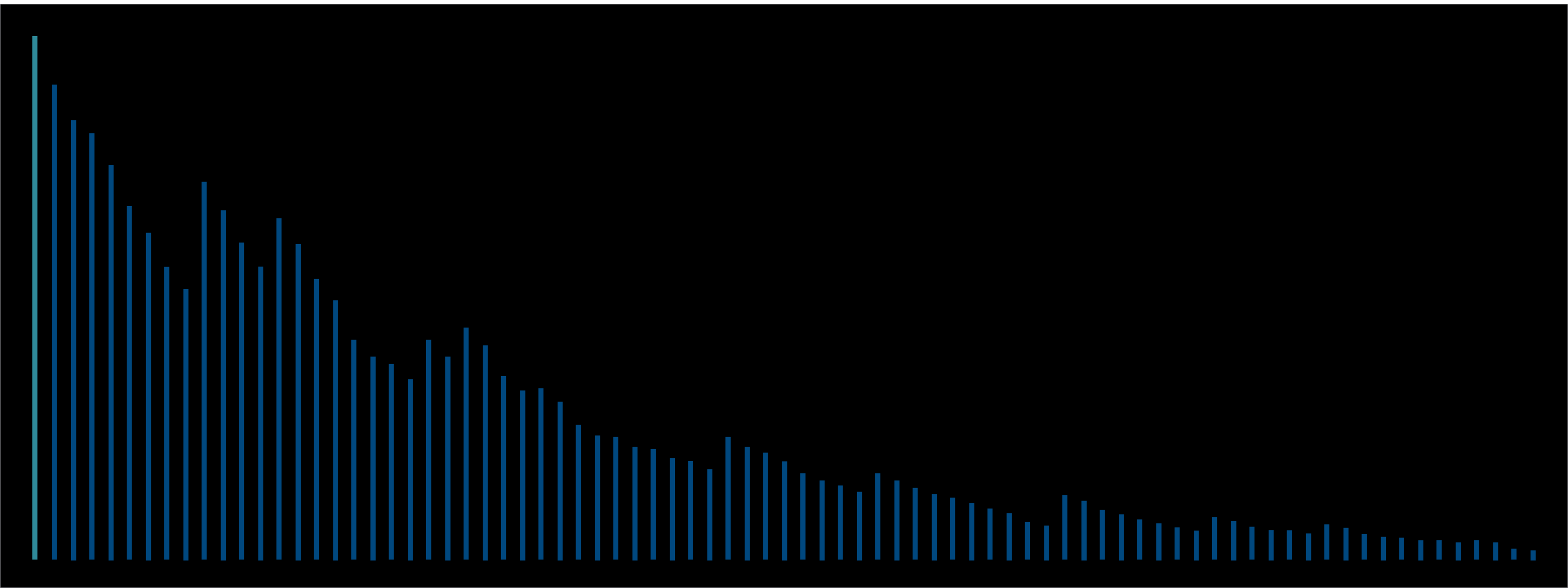
Security 38

Compliance 45

Conclusion 50

Authors 52

Research methods, Additional references 53



Executive Summary

Many enterprise organizations going through digital transformations are adopting DevOps as their new operating model for IT because their existing IT departments have proven unable to scale with business and customer needs. Microsoft defines DevOps as: "The union of people, process and technology to enable continuous delivery of value to customers."

DevOps unites software development (Dev) and IT operations (Ops) into teams aligned to business, product, and customer needs. This is a major departure from traditional IT practices, which are based on technical specialists—developers, testers, database administrators, and systems administrators—being viewed as their own independently operating silos. The end-goal of DevOps is to enable organizations to get product ideas to market faster, without sacrificing the stability or security of their production systems.

The practice of DevOps originated in tech-centric, digital-native organizations like Netflix, Spotify and Facebook, and these practices *evolved* with their technical architectures and organizational designs as the companies scaled. Most enterprise organizations, however, need to *transform* their existing people, processes and technologies in order adopt DevOps.

In April of 2020, McKinsey, looking to find tangible ways to evaluate how key business metrics were improved through software development innovation, created the concept of the Developer Velocity Index (DVI)[1]. DVI is a metric that "pinpoints the most critical factors (related to technology, working practices, and organizational enablement) in achieving Developer Velocity." Developer Velocity isn't just about increasing the speed of delivery, but is about unleashing developer ingenuity to solve complex business ideas—using new technology and ways of working to build software that meets the needs of enterprise customers, while accomplishing business goals. DVI is a composite metric of 46 key factors that influence Developer Velocity.

The key insights from the DVI study are as follows:

- **Developer Velocity Index (DVI)** is strongly correlated with organizational success. Companies with top-quartile DVI scores outperform bottom-quartile companies by 4-5x.
- **Developer tooling,** which covers collaboration, development, DevOps and planning tools, has the highest relative importance in driving business performance indicators.
- **Product management** cultures that utilize "psychological safety / fail fast and learn" philosophies, accentuated by product focused teams, achieve high DVI scores. Only ~20% of executives believe their organization implements this cultural practice.
- **Open source** capabilities and adoption is the most defining differentiator for top performing organizations trying to promote innovation and developer satisfaction.
- **Quality** is at risk in organizations with slow adoption of test automation practices and when companies believe that "quality" only pertains to testing.
- **Security** is a major concern, with 17% of organizations reporting they only test security vulnerabilities "for a major release" or "only when releasing to production".
- **Compliance** in enterprises is a major challenge, with 59% of respondents reporting that it can take "from multiple days to several months" to assess their current state of regulatory compliance.

After the DVI was established, Microsoft, in tandem with Sogeti, conducted a research study based on feedback from, and interviews with, Sogeti practitioners responsible for over 250 Cloud & DevOps implementations in enterprise organizations. The findings from the DVI study were combined with the Sogeti research to identify six key areas of enterprise-scale IT that face significant challenges as part of Enterprise DevOps transformation:

Product Management

Quality

Distributed Teams and Remote Working

Security

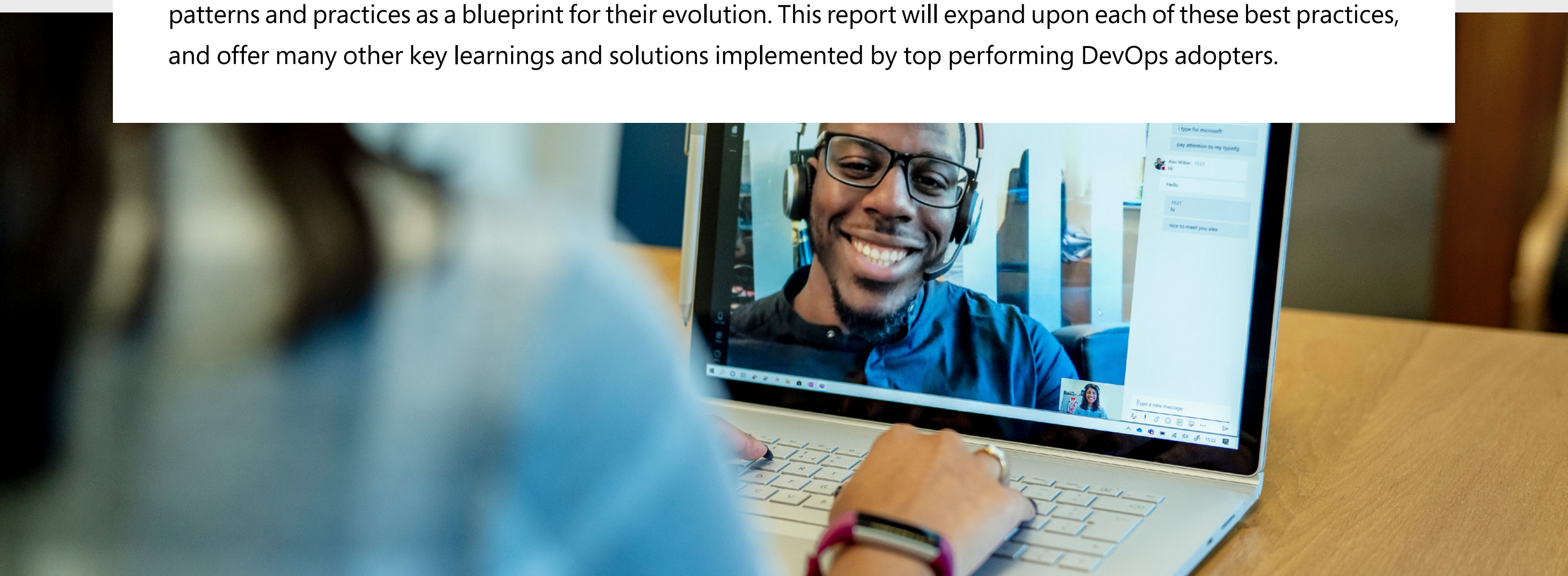
IT Governance

Compliance

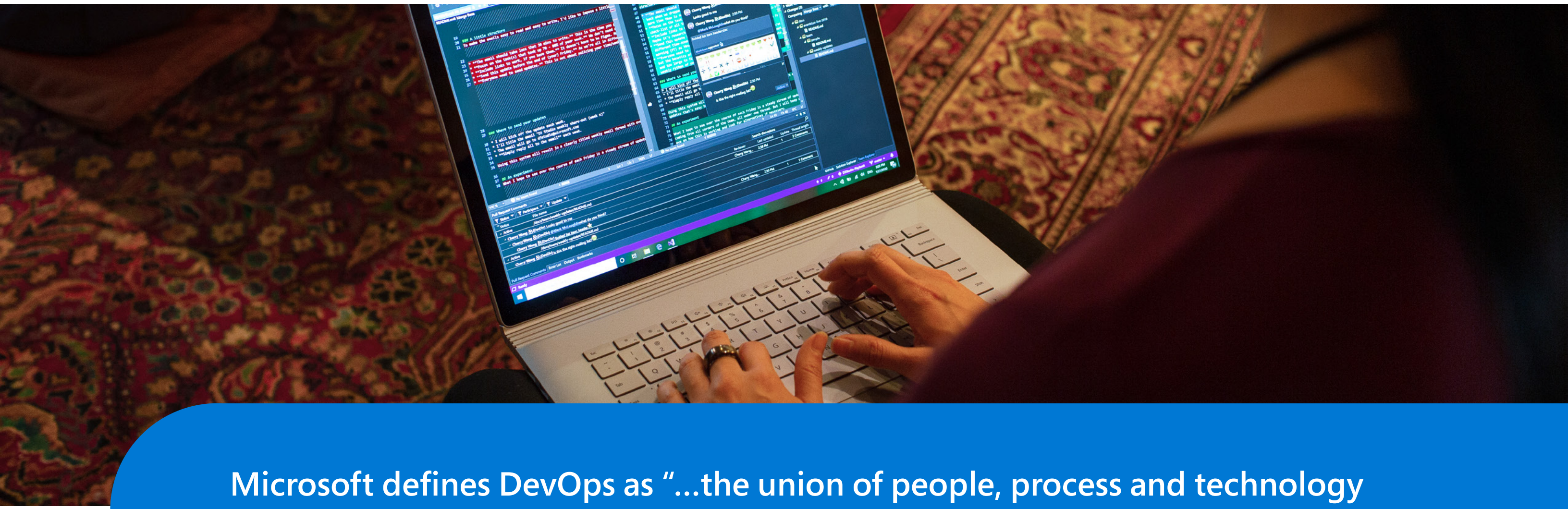
Across these six enterprise-scale areas, top performers in the DevOps space are addressing the challenges by utilizing the following practices and solutions:

- Moving from centralized "project-centric" to de-centralized "product-focused" delivery models, where DevOps product teams take full ownership of the end-to-end cycle of a product or service, while implementing iterative budgeting processes instead of large-scale annual budgets.
- Teams draw upon best practices from open-source projects and adopt an InnerSource methodology. These teams share reference architectures, application and infrastructure code, and common components to streamline and optimize their workstreams.
- Teams are being empowered to "code, ship, and collaborate from anywhere" by adopting cloud-based collaboration platforms, DevOps toolchains, and distributed version control systems.
- Enterprises are moving to principles-led models of governance, where governance and architecture teams define core principles, and outline how those principles will be assessed. These companies make doing the *right* thing, the *easy* thing, and complying with governance becomes a habit.
- DevOps practitioners are building, running, and managing their applications on self-service cloud platforms, and provide built-in security, compliance, and quality for all of their products.
- Organizations are adopting an "everything-as-code" model, where not only applications, but networks, compute infrastructure, security policy, build and release pipelines, etc… are written "as code" and stored in a source code repository. This practice enables continuous improvement by updating and improving on the source code, promoting better re-use across the organization and greater transparency.
- Enterprise-scale concerns like quality, security and compliance are moving from an "audit model" to a "continuous model," analogous to how the continuous integration/ continuous deployment (CI/CD) model has transformed software development. Quality, security and compliance are continuously assessed, compared to agreed standards and baselines, and deviations are corrected either via automation, or direct human intervention.

Organizations that are seeking to *evolve* towards an Enterprise DevOps delivery model can use these business patterns and practices as a blueprint for their evolution. This report will expand upon each of these best practices, and offer many other key learnings and solutions implemented by top performing DevOps adopters.



Introduction to Enterprise DevOps



Microsoft defines DevOps as "...the union of people, process and technology to enable continuous delivery of value to customers."

It's important to begin by acknowledging that the *goals* of DevOps remain the same regardless of whether DevOps is being implemented by a traditional enterprise or by a digital native organization. Both types of organization aspire to deploy to production faster and more frequently, and to provide highly available services for their customers in stable and secure environments.

Digital-native early adopters developed their DevOps principles via an *evolutionary* pathway—as they grew commercially to the scale they are at today, their DevOps practices were forced to evolve alongside their application architecture. In contrast, for many existing enterprise organizations, the reality is that they are stuck with years, or decades, of legacy applications and processes that struggle to remain coherent and relevant, while providing diminishing returns. These enterprises face a *transformational* DevOps evolution, and to succeed in such a dynamic and shifting technological landscape, they must begin by empowering their IT departments.

DevOps evolution has been largely inhibited by the way in which enterprises view their IT departments.

For many, IT is no longer fit for purpose. So, if executives want their IT departments to remain relevant—delivering quickly and flexibly—executives must fundamentally morph their conception of IT as-cost-center, and recognize it as a strategic business enabler that is pivotal to the success of the company at large.

The technological landscape has completely transformed and will continue to rapidly evolve in the coming years. DevOps is the best and most flexible solution for enabling enterprises to keep up with and anticipate next-generation IT. But, as the results of our research show, Enterprise DevOps adoption and transformation comes with a host of its own challenges.

Challenges of Enterprise DevOps Transformation

Product Management

Almost all enterprises have existing product/application portfolios, the majority of which are still hosted on-premises. Support for these software products is dispersed across fragmented teams, business units and organizations, leading to a lack of ownership, expensive duplication of systems and efforts, and disjointed customer support. Within these disparate product portfolios, enterprise organizations have often, out of hurried necessity, implemented patchwork solutions of varied commercial software that are incompatible with each other. This creates a culture that encourages the delivery of quick results, but often ends up causing catastrophic problems later on. These failures often come down to overburdened, undertrained product managers and team members who don't feel comfortable taking risks to solve systemic issues.

Distributed Teams and Remote Working

Enabling distributed teams and remote workers has become increasingly important in recent years, particularly so in the wake of the 2020 coronavirus pandemic. Nonetheless, previous challenges of how to enable developers to code, deploy, and collaborate from anywhere remain a constant. DevOps teams need to be equipped with the appropriate collaboration tools, high-speed network access, and the right computing equipment, all while maintaining secure environments. Creating and maintaining high-performance remote teams requires a combination of cultural and technological solutions.

Governance

Creating a governance framework that is effective at the speed of DevOps is a major hurdle for enterprises, particularly in the face of their diverse product portfolios and distributed teams. Existing governance models, with an emphasis on cost control and efficiency, are often counter-productive when applied to a DevOps model, which is more focused on value delivery and time to market. Instituting the right cultures within an organization to make "doing the right thing, the easy thing" is paramount in tackling many of these problems, where both cultural practices and business goals are clearly defined, with clear metrics to track the health of both.

Quality

Ensuring that quality is maintained as the frequency of software releases increases is a priority for Enterprise DevOps teams. This practice requires continuous quality engineering and a focus on how to balance the need for speed with a demand for quality. How do you move your DevOps teams from 'counting bugs' to actually improving the quality of a product? What software development lifecycle (SDLC) interdependencies do you need to be aware of to ensure quality—and how do you gain this insight, especially in today's open source environment? Breaking down the walls between different DevOps teams, standardizing how they work with policies and frameworks, building quality into CI/CD pipelines, and using automation to free up QA professionals and testing teams are all integral to addressing the quality within enterprises.

Security

The responsibility for security cannot rest with any one team inside the enterprise if distributed, autonomous DevOps teams are going to be empowered to build & run their own applications, and release changes at their own pace. Many enterprises even struggle to understand the vulnerabilities in their open-source code, and lack the ability to secure their cloud environments. Enterprises are able to eliminate a huge amount of potential bugs before they even commit code to field validation by adhering to secure SDLC principles, and automating processes. In order to do this, enterprises need clearly defined approaches to security, and to receive the proper investment and oversight at the board level.

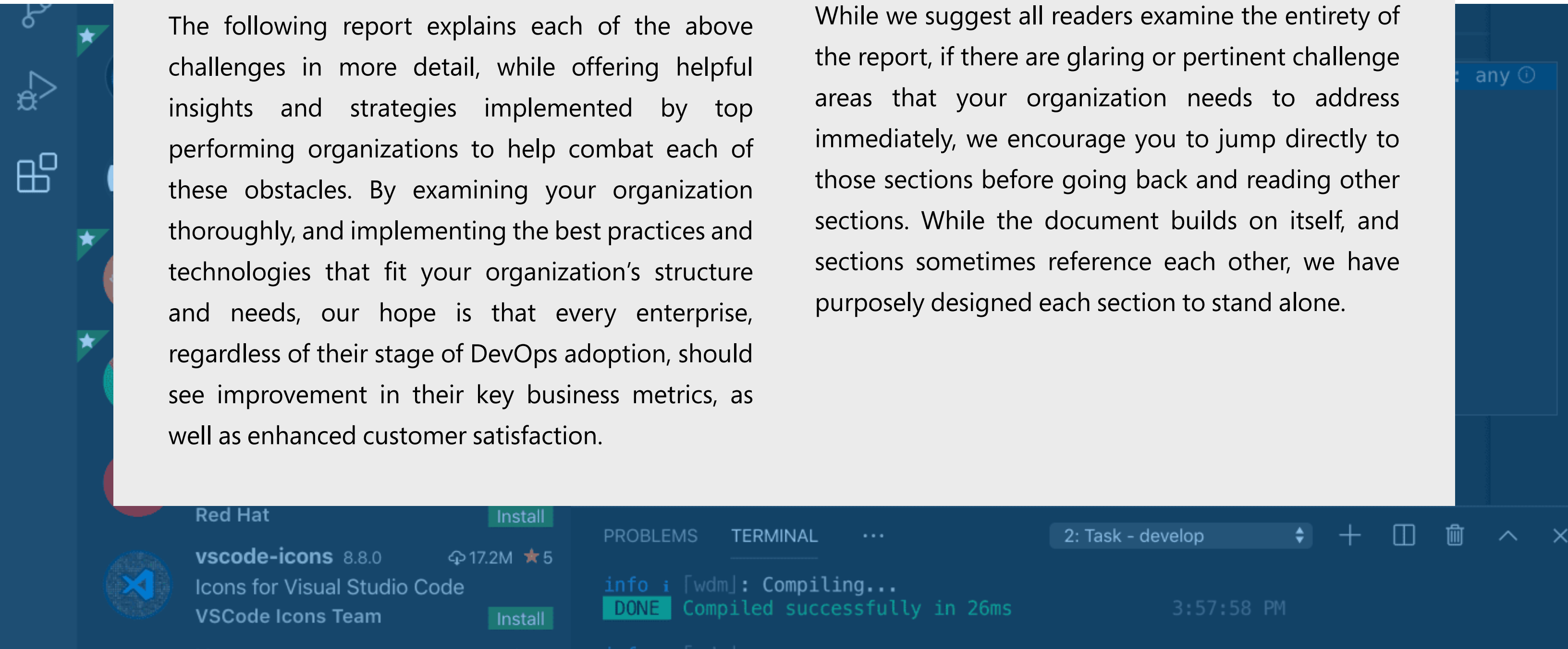
Compliance

Enterprise organizations are all subject to external regulatory oversight of some kind, e.g. SEC, GDPR, etc., as well as their own internal compliance functions. Ensuring that compliance is defined, maintained, improved, and constantly and consistently monitored is a critical concern and challenge for enterprises today.

A NOTE TO THE READER

The following report explains each of the above challenges in more detail, while offering helpful insights and strategies implemented by top performing organizations to help combat each of these obstacles. By examining your organization thoroughly, and implementing the best practices and technologies that fit your organization's structure and needs, our hope is that every enterprise, regardless of their stage of DevOps adoption, should see improvement in their key business metrics, as well as enhanced customer satisfaction.

While we suggest all readers examine the entirety of the report, if there are glaring or pertinent challenge areas that your organization needs to address immediately, we encourage you to jump directly to those sections before going back and reading other sections. While the document builds on itself, and sections sometimes reference each other, we have purposely designed each section to stand alone.



```
info ｢wdm｣: Compiling...
 DONE  Compiled successfully in 26ms                3:57:58 PM
```

The Developer Velocity Index (DVI)

Success in software development requires that organizations empower developers, enabling them to build productively, collaborate globally & securely, and scale innovation. Industry leaders have recently referred to this as Developer Velocity (DV). This goes beyond the traditional "agile" definition of velocity, as it's not just about speed, but about unlocking developer creativity to drive business outcomes. Just like velocity in physics, it's speed with direction.

With many potential improvement levers and little quantitative analysis, organizations often struggle with what to prioritize, and which development practices are the most effective.

Correlation analysis between the 46 factors that influence Developer Velocity (as shown in the below diagram) and business performance indicators revealed some striking findings about the critical drivers:

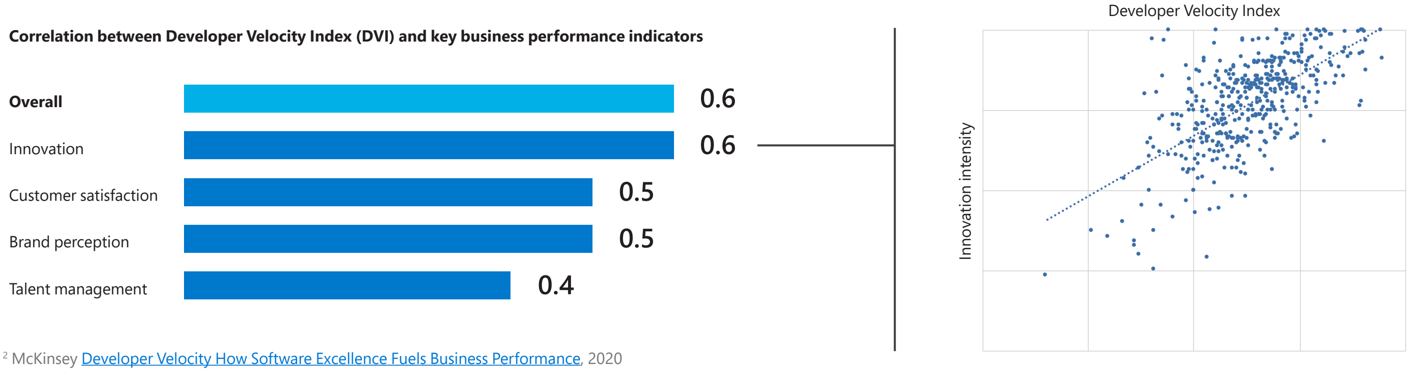
1. Only a relatively small group of critical drivers *actually* drove Developer Velocity
2. Numerous *perceived* drivers and best practices turned out to be less important than many executives believed.

When researchers compared organizations with top-quartile DVI scores to the bottom quartile, they found a four to five times improvement in Compounded Annual Growth Rate (CAGR)[2]. Looking beyond simple revenue growth, the top quartile DVI companies outperformed bottom quartile by 60% on Total Return to Shareholder (TRS) growth, and had a 20% higher operating margin.

Beyond purely financial metrics, DVI is also strongly correlated with other leading indicators of business performance, particularly innovation (See figure 2). The focus on business performance, rather than operational performance, is an important distinction for the purposes of this report. It suggests that top quartile DVI performers recognize the relationship between DevOps' combination of people, processes and technology as key drivers of business success, rather than focusing solely on traditional technology KPIs, such as total cost of ownership (TCO), systems availability, or change failure rate. Aligning DevOps transformation with successful business outcomes is what drives the cultural, technical and organizational changes reflected in this report.

TOP COMPANIES BY DEVELOPER VELOCITY HAVE AN INNOVATION EDGE

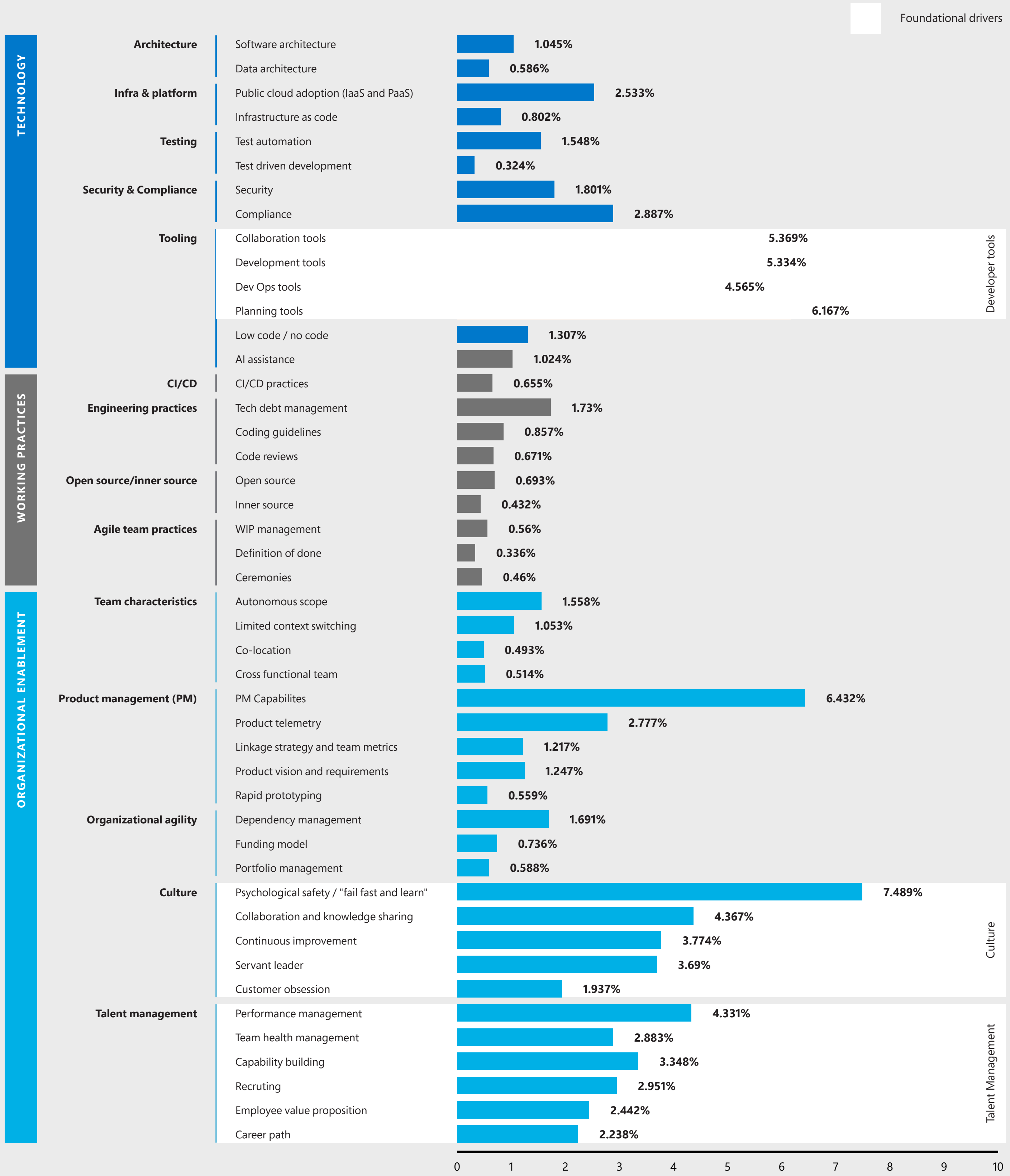
FIG. 1



[2] McKinsey Developer Velocity How Software Excellence Fuels Business Performance, 2020

DRIVERS RELATIVE IMPORTANCE ON BUSINESS PERFORMANCE INDICATORS

FIG. 2



Product Management

The 2020 McKinsey Developer Velocity Survey showed that strong product manager/owner capabilities is one of the top four factors correlated with a high Developer Velocity Index (DVI), as seen earlier. But why is a product-centric approach important to success for Enterprise DevOps?

DevOps advocates the formation of long-lived, multidisciplinary teams that are aligned to the delivery of value for the organization (value streams). Perhaps the most important practice is to create teams aligned to products in order to streamline efficiencies and create product-level accountability. These multidisciplinary product teams, comprising both Dev and Ops, take full ownership and accountability throughout the entire product lifecycle, from inception to retirement. This contrasts with the more fragmented, project-led approach of traditional IT that requires teams and individuals to take responsibility only for a narrowly defined segment of the product lifecycle—from Build to Release. They then hand the deliverable over to a separate Ops team for the operational lifecycle of the service. The consequence of this project mentality, as opposed to a product one, is a lack of overall accountability, which hinders long-term performance.

Shifting from "project" to "product" helps focus a team's attention and effort towards the product's goals, which must be measurable outcomes geared towards delivering customer value. Objectives and work backlogs are proactively managed to ensure that each product delivers maximum value (effectiveness) while maintaining low cost of ownership (efficiency). This fosters a dynamic and collaborative spirit where teams hold collective responsibility for success. Developers are more likely to ensure their code is up to standard if they work closely with the operations colleagues who will be directly impacted by any problems they cause. Furthermore, efficiency is improved through joined-up decision making and rapid feedback loops, avoiding the time-consuming cycle of escalation experienced by teams that have high levels of dependencies on other teams.

Over time, long-lived teams develop deep expertise and contextual knowledge about their product domain, which can be used to continuously improve and optimize their products for the benefit of the organization and its customers.

What are the challenges?

The results from the DVI survey provide a deeper understanding of the organizational enablement factors that drive KPIs. However, to start, it's important to delve into the cultural and talent management results and the role product-aligned teams play in driving business outcomes.

Trust and psychological safety

One of the key cultural findings concerns the contribution of "psychological safety" to business performance. Harvard Professor Amy Edmondson defines it as "a shared belief held by members of a team that the *team* is safe for interpersonal risk taking".

Inherent to this concept is the *team,* and with long-living, multiskilled, product-aligned Enterprise DevOps teams, there is ample room to take risks and innovate. The longevity of these teams enables trust to be nurtured and grown between team members. Trust within and between teams, and confidence about a product helps individuals feel safe to innovate, improve, and optimize while learning from their failures.

The culture and challenge management section of the DVI survey also revealed that collaboration and knowledge sharing are difficult without a clear bounded context regarding accountabilities and responsibilities of team members. Similarly, performance management is more complex unless the team can clearly see each member's contributions to the shared goals, and team members feel connected to and accountable for the work they do.

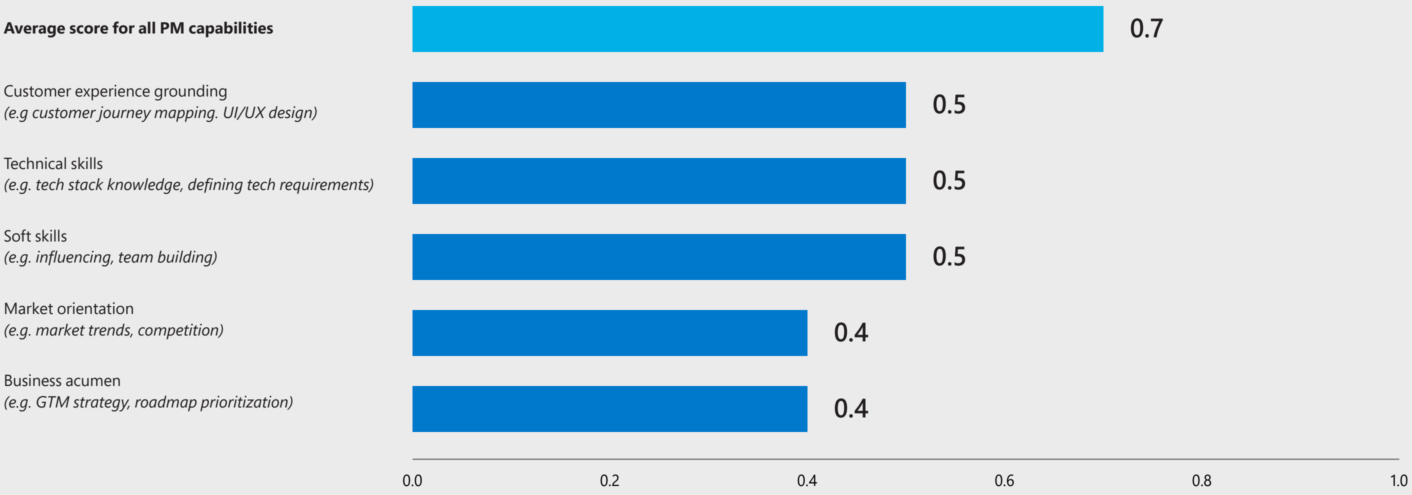
The product manager gap

When looking at the five main capabilities of a "strong, balanced product manager/owner", as shown in the survey results, we see that being a great product manager is no small feat. It requires a balance of emotional intelligence and soft skills, such as team building and strong customer-centricity. This should be complemented with good technical skills to be able to define the technical requirements, as well as the business acumen and market orientation needed to build the right products for success in the marketplace.

STRONG, BALANCED PRODUCT MANAGER/OWNER CAPABILITIES IS CRITICAL FOR BUSINESS SUCCESS

FIG. 3

STRENGTH OF CORRELATION BETWEEN PRODUCT MANAGER/OWNER CAPABILITIES AND BUSINESS PERFORMANCE



So, with the characteristics of a good product manager defined, surely it should be easy to build a product-centric team around strong product management? Unfortunately, that's not the case. One of the biggest challenges for Enterprise DevOps is that there aren't enough great product managers to go around. In fact, "CEOs and technology leaders often identify the role of product manager as one of their top talent priorities".[3]

As there aren't enough great product managers to lead the creation of innovative, in-house developed software solutions, many have turned to commercial-off-the-shelf (COTS) software to fill this gap. Again, unfortunately, while implementing COTS solutions can create a solid foundation to address enterprise business needs, they normally require customization and bespoke development to make them fit organizational requirements and needs. Thus begins the vicious cycle, as this integration and customization challenge raises the need for great product management.

Dealing with technical debt

One of the most frequently cited pain points in the Sogeti interviews was the concept of "technical debt." This is the accumulation of sub-optimal technical decisions made over the lifetime of an IT application, which creates situations where it becomes harder to make meaningful change and is frequently the "sand in the gears" that causes IT initiatives to grind to a halt. Many enterprise organizations have amassed large amounts of technical debt, where the cost of repayment of that debt can run into tens or hundreds of millions of dollars. Product-centric teams, as part of Enterprise DevOps adoption, alongside cloud migration, are forced to confront this accumulation, and subsequent repayment of technical debt.

Specifically, untangling legacy application architecture, the monolith application pattern, is one of the major obstacles when trying to create small product teams. This has led to the rise in popularity of micro-service architectures[4] to decompose colossal applications into smaller, more manageable components better aligned to a product team approach. It is important to note that while technical debt is a scourge on applications, it also affects processes and procedures. Enterprise DevOps transformation and adopting product-centric models offers organizations an opportunity to redesign workflows to streamline and accelerate the delivery of services.

[3] McKinsey, Insights- The Product Management Talent Dilemma, 2020
[4] Martin Fowler, Microservices Guide, 2020

Addressing the challenges

Create a product portfolio

The first step when implementing enterprise-grade product management as part of a DevOps transformation is to thoroughly assess your existing application portfolio and identify good candidates for early adoption of the product-centric approach. While doing this, top performers generally choose products that have fewer dependencies on other products. They also usually start with smaller products and work up. Limiting product team sizes will be important to begin with, and research into team performance recommends a team size of 7 ± 2. Starting small enables transformation leaders to bypass a large amount of organizational complexity, and helps set expectations for teams and roles.

When choosing to create a new product team, make sure to pick products that have measurable product value. Visualize product value (effectiveness) and costs (efficiency) for the lifecycle of the chosen product, and create a business value matrix to prioritize where your team is spending its time. It is worth mentioning that you do not need to exhaustively assess every application and service in your IT portfolio before starting your transition to a product-centric model. It is generally possible to quickly identify existing product/team alignments that can create quick wins to drive momentum for your transformation as you iteratively expand the product portfolio.

Fund your product portfolio

In many organizations, the way IT budgets are allocated inherently reinforces the divisions between Dev and Ops. Often the first step taken is to divide the money into a 'change' budget that's allocated to projects, and a 'run' budget, which is allocated to the support and maintenance of existing services, software licenses and other recurring operational expenses. Run budgets traditionally receive about two-thirds of annual IT expenditures, leaving products underfunded and under supported.

Furthermore, budgeting is generally an annual process, with funding allocated ahead of time for the next 12 months. This often leaves little scope for innovation and inhibits responding in an agile manner to emerging customer needs. Some organizations, led by the Beyond Budgeting movement, have gone so far as to dump the annual budget process entirely in favor of more flexible and adaptable budgeting methods.

Funding models that focus on the product and the product team are increasing in popularity. In these instances, funding is allocated to a product and, in turn, the product owner decides how to allocate that funding to support product teams in building and running the product.

Other organizations have embraced alternative portfolio funding models based on venture capital (VC) funding. In the VC funding model, investment is aligned to key stages in the product's development—seed funding to build a prototype and assess initial product/market fit, Series A funding to scale the idea, etc... Just as in a typical VC firm, an investment committee should meet regularly, usually every two weeks, to review new investment proposals from product managers and to assess the performance of existing products in the portfolio.

Support products with self-service platforms

As soon as a company moves beyond a small handful of product teams, it's beneficial to start introducing self-service platforms with core capabilities. With multiple autonomous teams, it's easy to unnecessarily reinvent core systems, such as CI/CD toolchains, ultimately wasting valuable time that could have been spent innovating. The job of these platform teams is to prevent separate product teams reinventing the wheel, and to provide capabilities that help them deliver their work.

Depending on your needs, you might have platform teams with mandates covering: continuous delivery, monitoring and observability, security and compliance, cloud infrastructure, test automation, and data.

In effect, platform teams are just product teams that provide a product to internal customers. It's important to think of the product delivery teams as customers because the platform teams should by and large have no mandate to control what the product teams do. This message is reinforced by Microsoft's own experiences. Microsoft were early adopters of the platform team model, forming the One Engineering System (1ES) team in 2014. The 1ES mandate was to empower every engineer in the company by standardizing on the best available tools. The1ES team has shared the lessons learned in their DevOps case study and their 5 steps to culture change whitepaper[5].

InnerSource

Another key part of the product and platform team model is the InnerSource (aka enterprise shared source) pattern. In their guide to adopting InnerSource, Danese Cooper and Klaas-Jan Stol define this pattern as: "A collaborative and empowering way of involving employees in making and implementing decisions throughout the corporation. It embodies a philosophy of human relations, an approach to rewards and motivations, and a loose, adaptable set of tools and practices."

Organizations with a large developer base are adopting InnerSource by following GitHub's core tenets of InnerSource[6]. Increasingly we're seeing this open, transparent approach applied to many other areas within IT, such as governance, reference architecture, security standards and best practices, but it can be applied in many other circumstances as well. The benefits that organizations ultimately attain from InnerSource adoption include increased delivery velocity, smoother collaboration between groups, higher-quality development, and better documentation.

Platform teams are crucial in this endeavor, as they provide transparency into how self-service platforms are built and maintained by sharing the source code via a code repository (repo). The platform teams should act as maintainers for this repo, but anyone can contribute to it, and if the product team finds they need to extend the capabilities of something a platform team has created, they shouldn't wait for the platform team to do it. Mirroring the contributor workflow of open source projects, the teams can code the change they want, then submit them back to the platform team as a pull request. Once it's reviewed and approved, the change can be merged into the main codebase. This whole process ultimately reduces churn, saves valuable time, and empowers employees to be proactive and take risks.

Cultivate product management talent

Cultivating and developing great product management talent within your organization is paramount to success. Given the competition in the marketplace for product managers, focusing on identifying and developing internal talent is the best strategy. To help companies achieve this daunting goal, McKinsey identified four key steps to build a world-class program for product management talent[7]:

1. **Articulate** the product management leadership development model for the organization.
2. **Provide** the product managers with organizational enablers for ongoing growth and apprenticeship.
3. **Leverage** a "field-and-forum approach" to design an end-to-end learning journey by blending on-the-job training with classroom or bootcamp learning.
4. Product manager recruitment should be a **strategic priority** for senior leadership.

[5] Microsoft Azure, Five Steps to Culture Change, 2019
[6] GitHub, Introduction to InnerSource, 2020
[7] McKinsey, Insights- The Product Management Talent Dilemma, 2020

PRODUCT MANAGEMENT CASE STUDY

In the fast-paced fashion industry, time-to-market is of the essence. But for one rapidly growing online fashion retailer, outdated and convoluted IT systems were struggling to keep pace with the company's booming business.

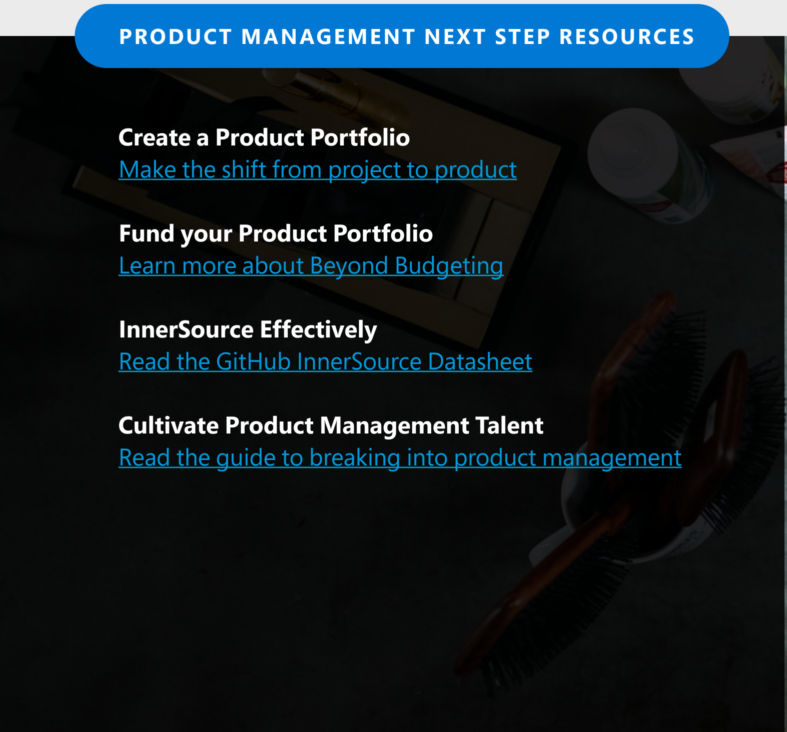
The retailer sells more than 80,000 branded and own-brand products via localized mobile and web experiences. Developing and maintaining all the applications needed to support their business had become a major barrier to innovation. IT was unable to scale beyond 300 releases per year, resulting in substantial pipeline bottlenecks. Maintenance was resource-intensive, while inconsistencies slowed down delivery and increased risk.

The number of development teams had escalated from two to 35 within a short timeframe, working across the retail website, payment solutions, ordering, and back office development systems. Software delivery was channeled through a complicated deployment solution, with a centralized deploy and release function. They desperately needed to modernize their systems to rapidly increase the speed and frequency with which software updates and new features could be released.

An Enterprise DevOps approach was adopted to replace complex, manual software delivery with new tooling and modern ways of working. Product-aligned Agile teams were created based on functional areas of the site, such as Search, or Shopping Basket, with the goal to empower each team to iterate and improve their product area as independently as possible. Each development team was migrated individually to a new continuous delivery pipeline, enabling them to build and deploy their products on-demand, while continuously innovating and taking risks.

The organization has now scaled to 80+ agile product teams, which release over 3,000 releases per year, enabling the organization to achieve a CAGR of 23% over the past four years.

PRODUCT MANAGEMENT NEXT STEP RESOURCES

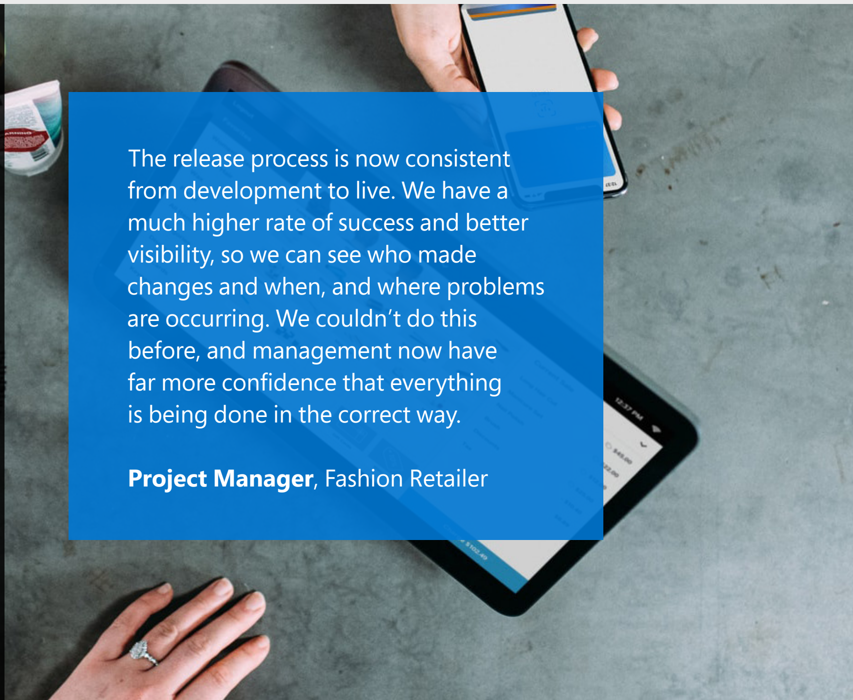


Create a Product Portfolio
Make the shift from project to product

Fund your Product Portfolio
Learn more about Beyond Budgeting

InnerSource Effectively
Read the GitHub InnerSource Datasheet

Cultivate Product Management Talent
Read the guide to breaking into product management



> The release process is now consistent from development to live. We have a much higher rate of success and better visibility, so we can see who made changes and when, and where problems are occurring. We couldn't do this before, and management now have far more confidence that everything is being done in the correct way.
>
> **Project Manager**, Fashion Retailer

Distributed Teams and Remote Working

Distributed and remote working is not a new concept for development teams—enterprises have offices around the world and need to provide capabilities for employees to work, and to support customers, in all time zones.

Many such organizations have outsourced work to increase their delivery capacity, while reducing costs. Some companies provide their employees with the flexibility to work from home, or any other location, whenever it suits them best. New streams of collaboration are starting to take place, both across company value chains and in the individual social domain of employees. At the same time, SaaS-based collaboration platforms bring advanced communications and document sharing tools within reach of every person in the enterprise. In particular, we've seen major advances in remote developer productivity tools.

Developer productivity platforms, like GitHub for example, enable distributed teams and remote workers to effectively collaborate to achieve their software development goals. Git, a distributed version control system originally developed for the Linux Kernel project, enables developers to work on source code disconnected from centralized repositories, but then easily merge their changes back into the central location. Understandably, different collaboration needs require specific support to function optimally. Many developers are highly experienced in remote work as a result of the unique composition of their industry, and this sees them collaborating across organizational boundaries on open source and InnerSource projects.

The COVID-19 pandemic introduced an urgent need to make the transition to remote work, with governments asking people to stay home and avoid going into their offices. This drastically changed ways of working and living. Team members, at all levels of responsibility, in organizations not deemed "mandatory" or "key" services have been required to work from home at some stage during the crisis.

Mandated remote working is vastly different than remote working as a strategic or operational choice. The speed at which organizations had to move to a fully-remote working model was a shock, and caused a frenzy within many IT departments that had no time to prepare. The changing economic environment meant that business models were forced to rapidly adapt, and additional emphasis was placed on efficiency and slashing costs, while somehow trying to provide the same value to customers as previously. Existing project and product roadmaps were largely cast aside. IT teams scrambled to do what they could, implementing video-conference capabilities, electronic signatures, and high-quality networks, all while provisioning hardware and developer environments, and making sure that company data was safe and secure.

As we continue to grapple with the idea of mandated remote working, we're presented with opportunities to learn from the challenges faced and to focus on growth. Although we can't be sure what's to come, to best prepare for the next phases of COVID19 and Enterprise DevOps adoption, several challenges must be addressed.

What are the Challenges?

Organizations are simultaneously having to adopt remote working practices, while adapting to a radically changed business landscape. Some companies are simply trying to survive, and are heavily focused on cost reduction, whilst other organizations are seeing a huge spike in demand for their services and need to work out how to meet that demand with a newly remote workforce.

From the perspective of DevOps teams, the challenges of running distributed teams can be categorized in three different areas:

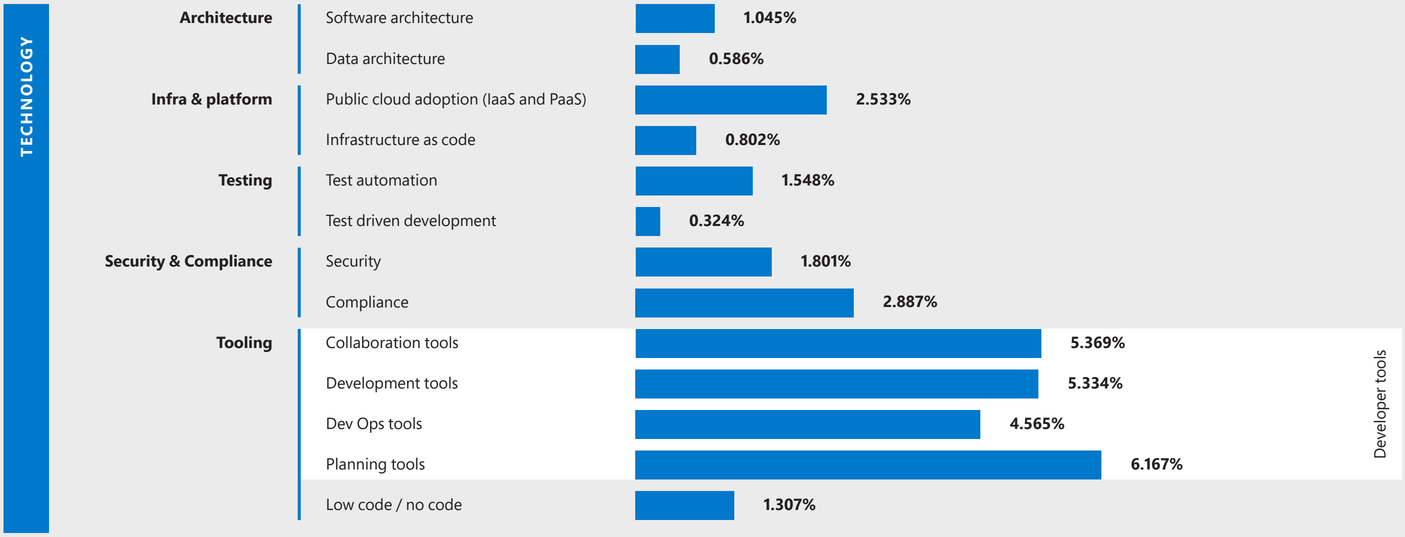
- How to **code** from anywhere—enabling developers to set up and maintain an effective remote-working capability
- How to **ship** from anywhere—build, test, and deploy application changes whilst continuing to manage production systems, securely
- How to **collaborate** from anywhere—effective distributed team working including voice, code and document sharing

Remote-working capability challenges

As the DVI survey indicates, having the right developer tools is one of the largest drivers of business performance, and collaboration tools are one of the key factors in that toolchain. It's clear that many companies have underinvested in creating toolchains to support remote development and collaboration, leading to a scramble to adopt patchwork solutions that will have major repercussions later on.

DRIVERS RELATIVE IMPORTANCE ON BUSINESS PERFORMANCE INDICATORS

FIG. 4



50%

increase in bugs for teams that are distributed vs. co-located[8]

Building effective toolchains typically takes time and investment. However, the COVID-19 pandemic has resulted in many organizations massively accelerating the rollout of their developer and remote collaboration toolchain, and companies are already reeling from the downstream consequences. These range from seemingly small problems like a lack of computers and not enough bandwidth, to more complex and worrisome challenges, such as insecure physical environments that are unsuitable for confidential or classified work, and a big increase in bugs[8].

41%

spike in exposed remote desktop connections as teams try to access work environments remotely[9]

Enabling team members to remotely work in a frictionless way is a major challenge. Not being able to track work, track progress, compile code, run tests, or simply do the jobs that need to be done becomes a struggle without the right tools and the ability to run them remotely.

Security, when working in a distributed manner, has multiple layers—network connections need to be secured and encrypted, while access to company or customer assets must also be secured. In many organizations, company data is not even permitted on personal environments. Managing source control tools like Git, file replication systems like OneDrive, and other tools that clone and synchronize data seem to offer potential solutions, but also bring up a host of their own problems.

Team collaboration on assets from anywhere

Team collaboration is about more than just being able to allocate work items to team members. To innovate, team members need to be able to collect customer feedback, share ideas, brainstorm and solve problems together - tasks that are made even harder with a fully remote workforce. Organizations need to equip teams with the right remote collaboration tools and train them to work efficiently within the limitations imposed by remote working.

Deploy from anywhere

Organizations with large on-premise application portfolios, that are being managed by on-prem management tools, are facing significant challenges now that their system administrators are forced to work remotely. To maintain and update these on-premises systems, a secure remote-access solution must be available, with the capacity to handle the increased volume to remote workers. Typically the design and implementation of a secure remote access solution, or to scale it's capacity, takes time - time many organizations don't have.

[8] McKinsey, A New Management Science for Technology Delivery, 2019
[9] ZDNet, RDP and VPN Use Skyrocketed Since Coronavirus Onset, 2020

Addressing the Challenges

Collaboration platforms

Remote collaboration tools, like Microsoft® Teams and Zoom, have been quickly adopted by all organizations shifting to fully remote work. And, while existing DevOps teams were largely familiar with online collaboration tools, others took some time to adjust. Most DevOps teams are familiar with distributed version control systems, making the transition seamless, but to stay connected, many teams have set up multiple team checkpoints to keep team participation high.

DevOps teams require informal connections, to imitate less organized knowledge sharing and collaboration, which is why they set up chat channels in Slack and Microsoft Teams as a substitute for desk drop-ins and coffee chats. Peer reviews, design brainstorms, etc. require different approaches. For example, Git collaboration capabilities like "Pull Requests" and the capabilities of the supporting tools seamlessly enable developers to add peer reviews and comments when working remotely.

For training scenarios and ad-hoc explaining, successful teams and organizations are adopting Visual Studio Live Share, enabling developers to code together in the same environment. The same toolset can be used for training, or for hiring interviews, as it provides a collaborative space for developers to work together, side-by-side, virtually.

Cloud as a remote working enabler

Enabling teams to securely access code and data has been one of the biggest hurdles for IT departments to overcome as the workforce was forced to go remote. In particular, many developers might not have access at home to the high-spec computer hardware they enjoy in the office environment, nor the secure access to development environments and data.

Cloud solutions have helped businesses overcome this problem. To alleviate low performing and insecure local computers, top performing organizations have turned to Virtual Desktop Infrastructure (VDI) solutions or Virtual Machines (VMs) in the cloud. A centrally-managed, cloud-hosted developer environment can help solve many of the environment setup issues developers frequently encounter. Furthermore, organizations that had previously adopted SaaS-based DevOps platforms, like GitHub or Azure DevOps, had already solved these challenges, and team members were able to seamlessly transition to remote working.

{ }

2 days

typical time it takes developers to set up an environment[10]

Organizations already leveraging cloud environments (as opposed to on-prem environments) for test scenarios could simply keep on executing test cases. On the other hand, teams with no cloud testing capability struggled mightily with test execution and turned to services like Azure DevTest Labs to alleviate this pain.

Based on the short survey relating to 250 Sogeti client projects, we determined that the maturity of DevOps per se did not influence the remote work transition significantly, although there was a small difference in ease of transition for more DevOps mature teams. However, the adoption of, and investment in, cloud platforms did simplify the transition to remote working, largely due to the scaling capabilities and the fact that secure remote access solutions were already in place. This meant that these enterprises could keep on working and seamlessly scale to a more distributed organization.

[10] Internal Microsoft survey results

REMOTE AND DISTRIBUTED WORK – BEST PRACTICES FROM MICROSOFT

The following comes directly from an internal Microsoft communication setting out specific areas of focus to enable remote productivity of its developer workforce during COVID-19.

Encouraging team culture

Working remotely limits our ability to pop over to someone's desk, walk the halls, and gather for group lunches. This makes it all too easy for team members to become disconnected. To retain a collaborative team environment, we have established team practices, including:

- A culture channel in Microsoft Teams to use for virtual "water cooler" hangouts.
- Virtual lunches and coffee breaks.
- Outdoor walks during check-in calls with teammates.
- A conference call bingo board, with things like "dog barking" "kid wanders in" etc.
- Use of chat for live questions and comments and team cohesion, particularly when some folk may need to keep themselves on mute because of their work environment.

Staying informed

Keeping employees informed is a key aspect of remote work.

- Our team created an online list of helpful information—from remote-work pointers and company-wide guidance to current hardware and tips for self-hosting. It's updated daily, if not hourly, with the latest information as we get it.
- We also try to record virtual events, and more day-to-day meetings in general, so that people can catch up and watch on their own schedule or time zone.

Tech setup

To optimize our engineers for remote work we start with their tech setup, making sure they have the right hardware and network connections. Most of our engineers already have a Microsoft-provided laptop to enable them to perform their duties remotely. We've also encouraged employees to bring monitors, hardware, and peripherals to mimic their work configurations in the office. This is especially important for developers who use two- or three-monitor configurations.

One of the remote-access learnings for development teams is, of course, that cloud services can be a critical infrastructural option for enabling remote developer productivity. For example, GitHub Codespaces allows you to create cloud-hosted dev environments, and then connect to them from editors of your choice.

Even for developers who work exclusively on desktops, the cloud can help. They can use corporate laptops with a Windows Virtual Desktop solution that allows them to remote into their dev environment running on powerful VMs in the cloud vs. locally on their device.

Shipping

Next, we need to maintain our well-established rhythms for shipping software. Our engineering teams rely on regular team stand-up and ship-room meetings to keep everyone moving forward. These meetings can't—and don't—stop.

Instead, we've moved them exclusively to Microsoft Teams. In addition to being the hub for remote meetings, Teams has apps specific to software development, such as Azure Boards for planning and tracking work, and Microsoft Whiteboard, which lets meeting participants sketch out flows and map ideas. These tools help bring our team members together as though they were in the office.

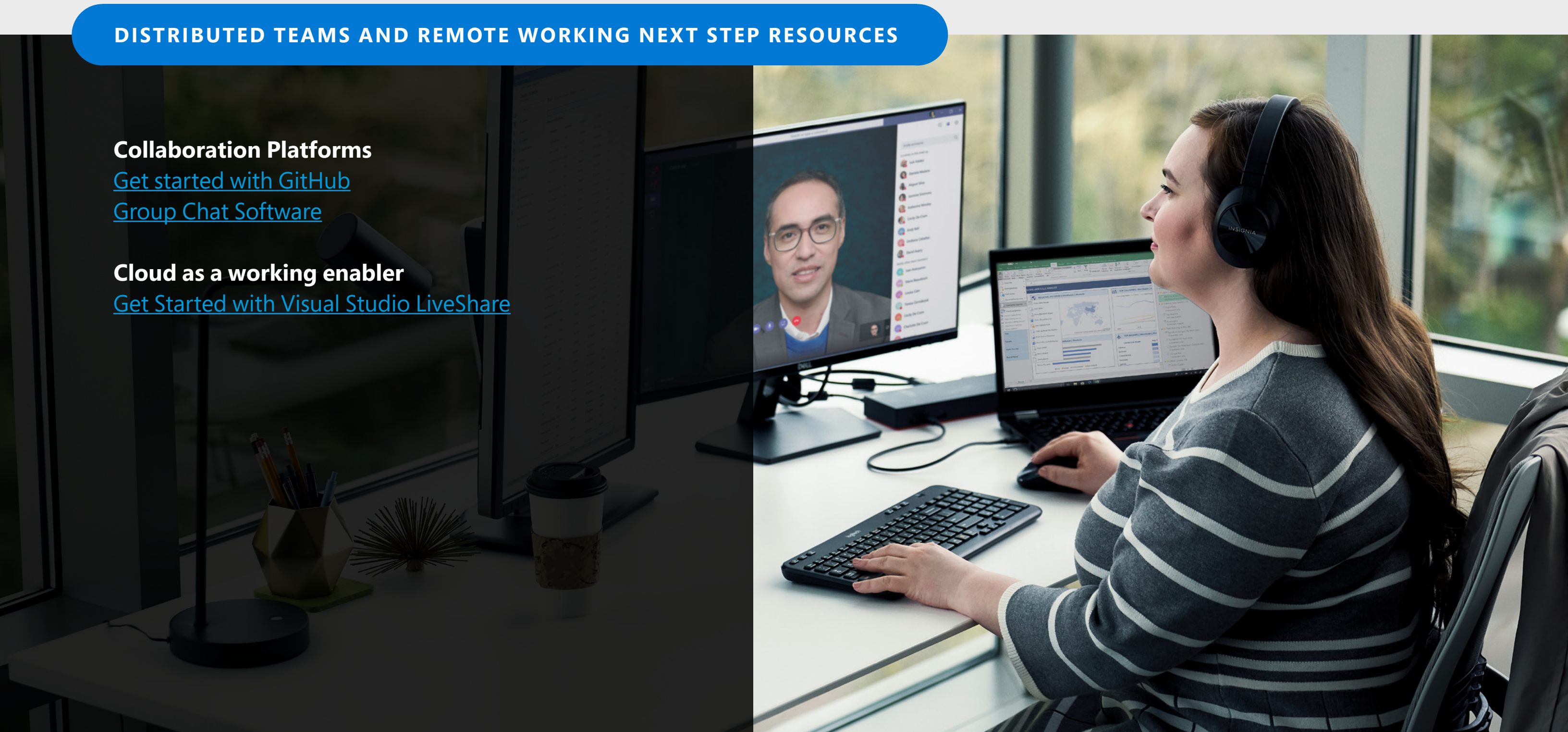
Peer learning

Working remotely can be a challenge for developers who participate in peer reviews or pair programming, where they would normally sit side-by-side and learn from one another. Our developers use Visual Studio Live Share for joint debugging sessions and peer learning. Live Share allows developers to work both together and independently, and feels a lot like in-person collaboration.

Hiring

Hiring is another essential process that can't be put on hold while we work remotely. We've found it helpful to use Live Share for technical interviews, allowing us to engage and communicate with candidates using tools they're already comfortable with.

DISTRIBUTED TEAMS AND REMOTE WORKING NEXT STEP RESOURCES



Collaboration Platforms
Get started with GitHub
Group Chat Software

Cloud as a working enabler
Get Started with Visual Studio LiveShare

Governance

IT governance is part of the overall enterprise governance strategy and ensures IT decisions are aligned with the business strategy. Good governance is about not only 'doing the right things' but also 'doing things right'.

There is a wide breadth of IT governance frameworks; from more common ones like COBIT 2019, to international standards like ISO38500. Most frameworks describe the roles, responsibilities, organizational structure, and processes to guide the decision-making within IT.

Governance influences how goals are set and achieved by IT. At the same time, governance also conflicts with the independence desired by DevOps teams. These teams need to follow the organization's standards in terms of architecture, security, and procedures, but often see them as obstacles to rapid delivery. While they are guiderails to support them, the team often views them as innovation inhibitors. But these guiderails can also be the base of knowledge sharing to grow teams and individuals. This tension between the chaffing restrictions of traditional governance models and the broader benefits that good governance can bring is at the heart of the DevOps governance challenge.

Startups and small agile companies tend to lend themselves well to adopting and maturing DevOps practices. They often have lightweight governance processes that are easy to manage and even easier to automate. Larger enterprises with deeply rooted governance processes and strict rules often take longer but can benefit hugely from simplification and automation.

If IT governance often determines the speed of IT change, then it can either be a force for enabling rapid change, or slow down value delivery.

What are the Challenges?

IT as cost center rather than a strategic enabler

Many existing governance implementations are based on the IT-as-cost-center model and, as a result, their primary focus is on cost effectiveness. DevOps, on the other hand, is based on an "IT-as-strategic-business-enabler" model, with a dedication to accelerating innovation and time-to-market. The sluggish pace of existing non-DevOps governance processes cripples innovation, and often makes IT feel more of a hindrance than an asset.

One large financial services corporation we interviewed takes a minimum of two months and 23 handovers to get a purchase order approved. In another example, a global telecommunications provider takes a minimum of four weeks for a proposal to be approved for new software development, five weeks to obtain an environment, two to three weeks to make a firewall change, and four to six weeks for end-to-end testing. Each of these steps is required to pass governance approval processes via service desk tickets being raised. These processes are purportedly in place to "save money."

Overly prescriptive governance

Highly prescriptive rules-based governance frameworks break down at enterprise scale. Across an enterprise's technology landscape, with different operating systems, polyglot software implementations, and diverse attitudes to risk, the number of times an exception to the rules is required ends up resulting in *everything* becoming an exception, completely undermining the goals of good governance.

Legacy governance implementations largely struggle to adapt to the rapid evolution of technology. For example, if we look at the evolution of application hosting—from physical servers, to virtual servers, to cloud hosted VM, to containers, to serverless—each evolutionary step needs governance rules. The pace at which technology is evolving means that trying to prescribe governance at a level of detail tied to technology implementation, while ensuring it remains relevant, is virtually impossible.

Making it about more than just IT

A popular perception is that DevOps is an 'IT thing' or a 'developer's toolkit', like Agile was before it. In reality, while DevOps is a newer operating model for IT, it is intrinsically linked to the entire organization's digital transformation and the governance framework needs to encompass the whole company, not just IT. However, the DevOps model itself poses governance challenges. Too often DevOps ways of working, notably the freedom and autonomy granted to teams, has been misinterpreted by some teams as a freedom from process or any formal governance—a misconception that needs to be addressed during implementation.

The need for real-time intelligence

Finally, without effective feedback, ideally in real-time, it can be difficult to achieve appropriate levels of governance. Dynamic application environments, the rapid release of new features, and the faster adoption of new technologies mean that the need for real-time business intelligence that underpins the governance model is more important than ever.

Addressing the challenges

Make doing the *right* thing, the *easy* thing

Successful governance teams should aim to work towards an overarching governance principle that makes doing the *right* thing, the *easy* thing. In this way, complying with governance becomes the path of least resistance, rather than a hindrance or obstacle to overcome.

The governance function can support DevOps teams by publishing best practices that provide helpful, actionable, guidance. Building a knowledge sharing culture can also make the whole organization more efficient and help grow teams and individuals in terms of their experience and capabilities.

Principles, not rules

Tichaona Zororo, a director at ISACA, the organization that publishes the COBIT governance framework, points to a key tenet when he says: "Governance should be principles and outcome based not rule based".

Governance is *not* about adopting a command and control mindset that seeks to mandate rigid rules to control the delivery of IT services in a rapidly evolving IT world. Rather, it is about outlining a set of core principles or outcomes to which teams and products are meant to adhere. Governance teams should provide common patterns, frameworks and tools that assist teams in adhering to these standards, while clearly delineating expectations. This, in turn, gives teams the flexibility to develop their own implementations, so long as they align with the agreed upon principles and outcomes.

An excellent example of this approach is the UK Government Digital Service Standard designed to improve the delivery of digital services by UK public sector organizations. The standard outlines 14 high-level principles that govern the full lifecycle of a digital service—see The Service Standard helps teams to create and run great public services.

Before any given service goes live, teams need to demonstrate that their service aligns with the defined standards, even though the specifics of each team's implementation may vary. For example, Standard #7 states, "Use agile ways of working" but doesn't mandate Scrum, DSDM, Kanban, or any other particular agile method. Implementation methods are left to each team, although whatever the team chooses must align with the more detailed guidance.

ADOPT AN INNERSOURCE MODEL

According to GitHub's Core Tenets of InnerSource[11], successful governance implementation should be based on these five key principles:

Open
Democratizing access, creating a level playing field for the open sharing of work, ideas, and feedback, and ensuring cultural and strategic alignment.

Transparent
Ensuring the process as well as the product is visible, predominantly by decoupling communications from time and space.

Participative
Sharing work and making it easy for others to discover, use, and contribute.

Collaborative
Working together to incrementally increase quality, distribution of knowledge, and shipping velocity.

Governed
Directing, guiding, and supporting the software community, through standards, patterns, roles, and executive sponsorship.

Governance should be viewed less as an external process imposed on teams and more a collection of shared best practices that makes work better and more efficient.

At a very practical level, ensuring everything (from application source code and architectural patterns, to security standards and development methodologies) is both open and transparent makes it evident that anyone has the opportunity to challenge these standards and contribute to improving them.

Governance guidelines and artifacts maintained via an InnerSource model bring shared ownership to teams and an increased drive to use and follow the guiderails of governance.

[11] GitHub, InnerSource, 2019

Platforms & executable reference architectures

Top performing organizations are creating self-service technology platforms, supported by comprehensive documentation that helps users leverage these platforms to ultimately build better products. These shared platforms enable organizations to embed and facilitate governance, while making it easy for teams to do the right thing.

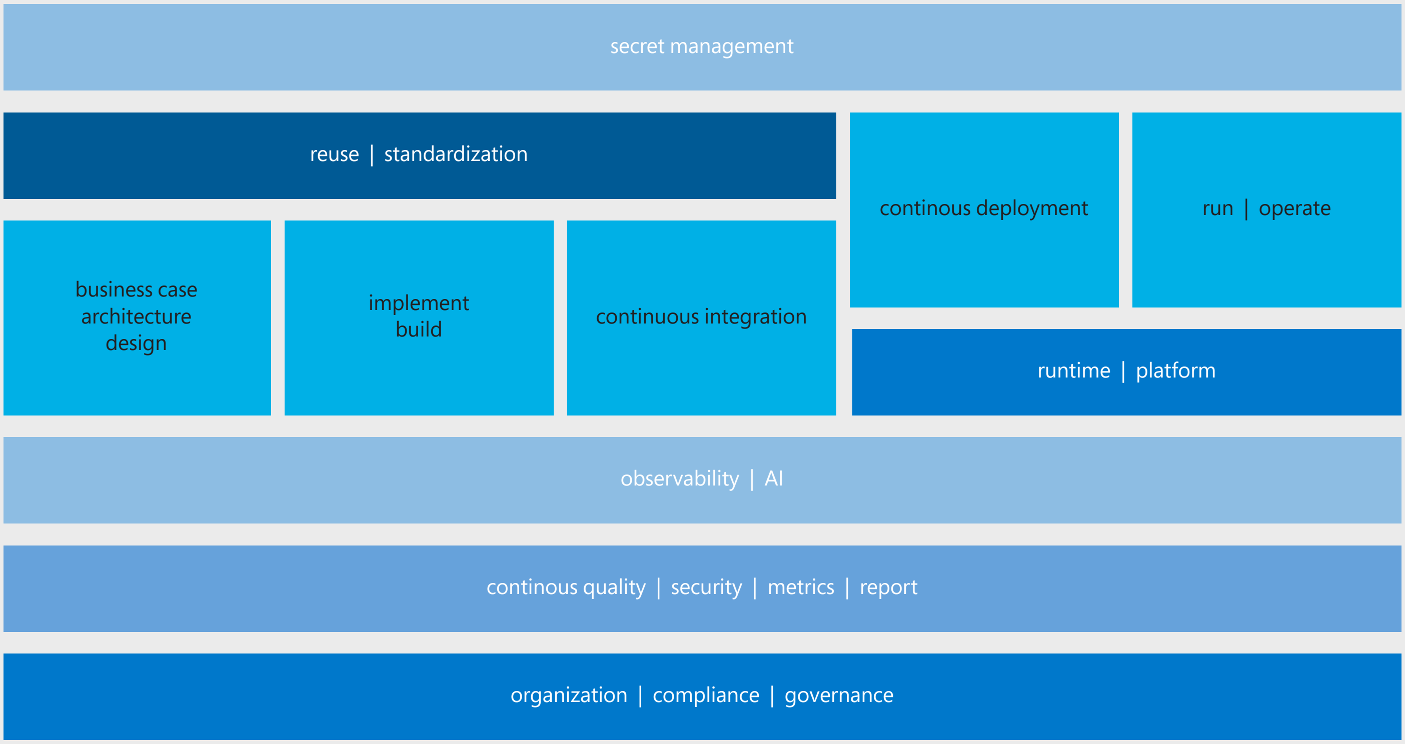
A company platform might include security, identity, consistency, cost management and deployment automation as key services provided. Each of these self-service platforms will have governance "built-in", so that teams who use the platform inherit the governance best practice at little or no effort to themselves. As an example, one way of ensuring that cost is accurately tracked and attributed to the correct products or departments is to utilize resource labels. The use of these cost management features can be mandated in the company platform.

Consistency at enterprise scale is often a governance challenge. Organizations can have hundreds of developers working in self-organizing teams, so it is important to have consistent implementation practices across the IT landscape. Almost all enterprises make use of reference architectures with best practice guidance for how to build and run applications, as well as how to incorporate governance and security. One of the more mature governance practices is to make this reference architecture executable via sets of templates, scripts and tools, and to move the reference architectures beyond Visio diagrams and Word documents into code when combined with DevOps tools like infrastructure-as-code, and configuration-as-code.

When combined with the InnerSourcing patterns mentioned previously, teams can contribute to the creation and constant iteration of these reference-architectures-as-code, helping to ensure a system of dynamic and effective governance.

INTERNAL SOGETI RESOURCE LABELING DIAGRAM

FIG. 5



Continuous governance

Traditional approaches to governance have largely relied on an inspection process or quality gate prior to production/release, which isn't feasible at scale and pace. As a result, top performing companies are adopting continuous-delivery models for governance so that they can deliver value to customers sooner via the frequent release of new software features.

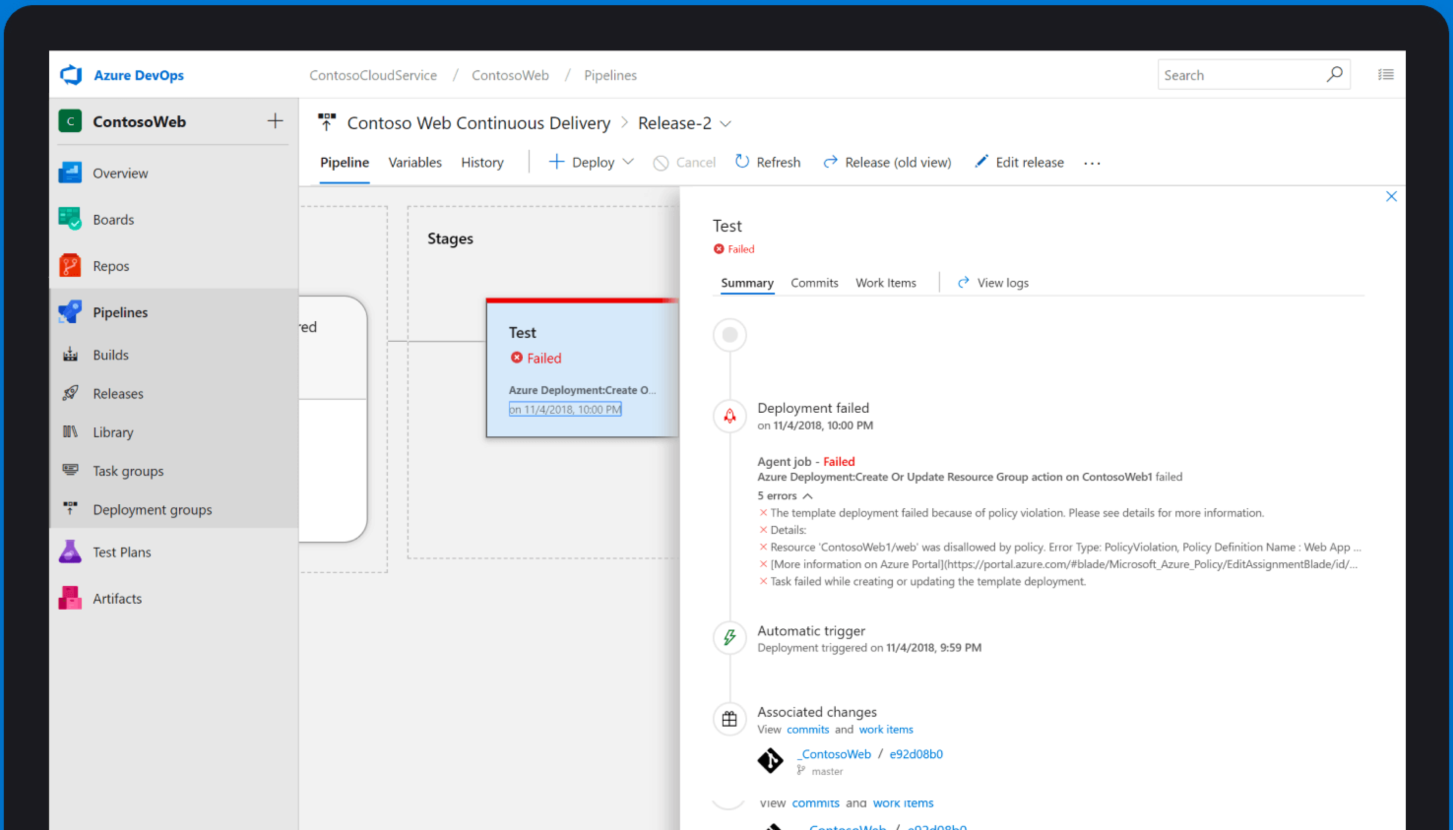
To combat cumbersome governance practices, enterprises are also moving towards automating their governance, reducing the burden on IT departments that no longer need to perform sluggish approval and validation workflows. At the heart of this continuous approach are CICD pipelines with governance steps built into each pipeline, which enable the frequent release of secure and high-quality products to customers. Continuous governance defines the principles that can be implemented via automated software controls that promote code from one stage of the pipeline to the next.

Seek feedback and insight

Traditionally, it has been difficult to gain visibility into the current state of an organization's adherence to its governance principles. It's also been hard to measure the organizational benefits of following that governance.

Adopting Enterprise DevOps solves this. One of the key tenets of DevOps is to create effective feedback loops—the faster the feedback cycle, the quicker an organization can learn what works and adapt accordingly.

Enterprises are increasingly turning to AI and machine learning platforms to solve this challenge. Governance teams use this technology to filter out and correlate data in order to turn the large volume of data generated by DevOps toolchains and cloud platforms into actionable intelligence that they can use to optimize their governance processes. By filtering out and correlating data, teams can focus on taking the corrective actions needed without being bogged down by the sheer amount of data that has been collected.



Deployment failed
on 11/4/2018, 10:00 PM

Agent job - Failed
Azure Deployment:Create Or Update Resource Group action on ContosoWeb1 failed
5 errors
× The template deployment failed because of policy violation. Please see details for more information.
× Details:
× Resource 'ContosoWeb1/web' was disallowed by policy. Error Type: PolicyViolation, Policy Definition Name : Web App ...
× [More information on Azure Portal](https://portal.azure.com/#blade/Microsoft_Azure_Policy/EditAssignmentBlade/id/...
× Task failed while creating or updating the template deployment.

Automatic trigger
Deployment triggered on 11/4/2018, 9:59 PM

Associated changes
View commits and work items
_ContosoWeb / e92d08b0
master

View commits and work items

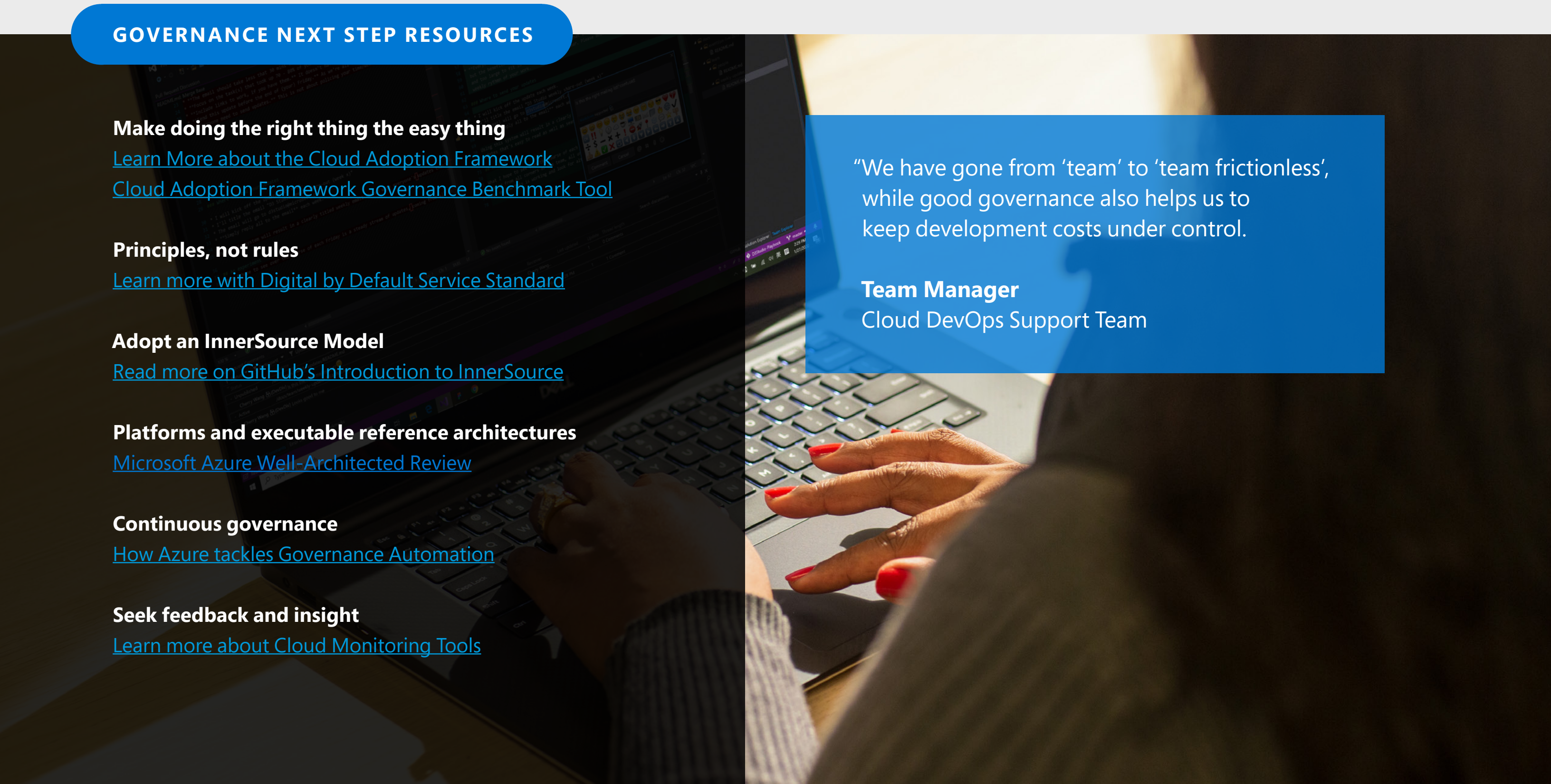
GOVERNANCE CASE STUDY

For one large European insurance company, good governance is critical in avoiding what it calls “chaos and security issues” across more than 20 different technology and development streams, each with their own domain architectures and DevOps teams. With over 700 active DevOps professionals, effective governance ensures that these teams work consistently, while complying with company and regulatory dictates—all without impeding speed of development.

To embed good governance, the insurer implemented a central team that monitors and supports adherence to the development toolchains. In turn, these toolchains have been rationalized with the help of enterprise architects and domain architects to bring more control and efficiency, and any disruptions, security issues, or illegal usages of tools can be quickly fixed.

As DevOps is as much about the people as it is about the enabling technologies, communication is vital in the firm’s approach to governance. An IT service portal provides information on offerings and services for teams, together with a complete set of ‘how to’ and other relevant documentation. The communications team gives oversight into current happenings and future plans, while a communication forum is used for knowledge sharing between team members, which greatly reduces friction and work disruptions.

This governance team is highly active and keeps up its momentum via a notification and updates channel, as well as with regular mail distributions. For the insurer, this commitment to good practices through governance ensures that its DevOps teams work in tandem to support the business goals efficiently and compliantly.



GOVERNANCE NEXT STEP RESOURCES

Make doing the right thing the easy thing
Learn More about the Cloud Adoption Framework
Cloud Adoption Framework Governance Benchmark Tool

Principles, not rules
Learn more with Digital by Default Service Standard

Adopt an InnerSource Model
Read more on GitHub’s Introduction to InnerSource

Platforms and executable reference architectures
Microsoft Azure Well-Architected Review

Continuous governance
How Azure tackles Governance Automation

Seek feedback and insight
Learn more about Cloud Monitoring Tools

> “We have gone from ‘team’ to ‘team frictionless’, while good governance also helps us to keep development costs under control.
>
> **Team Manager**
> Cloud DevOps Support Team

Quality

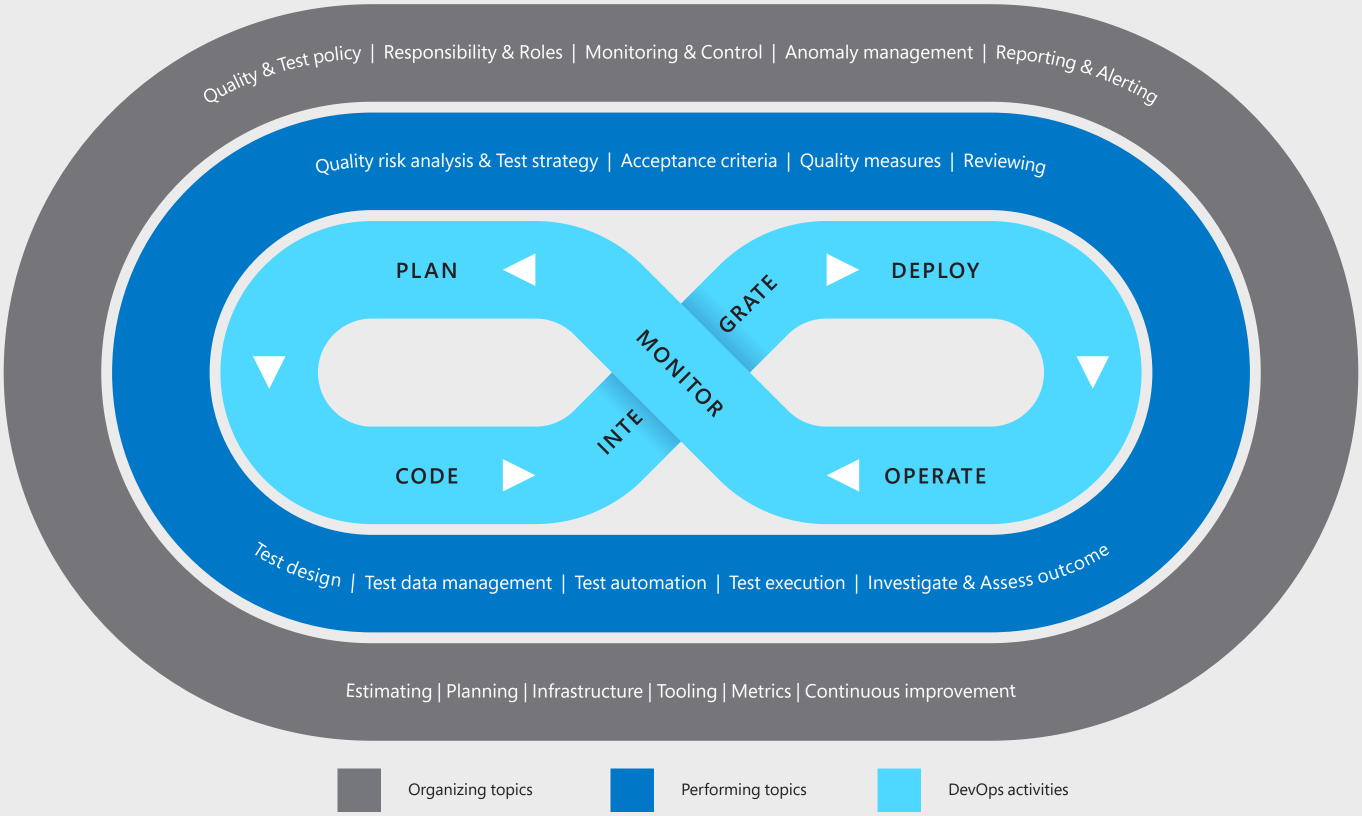
Sogeti defines quality as “the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs[13].”

Achieving this requires quality engineering as a discipline, which enables DevOps teams to deliver business value with high quality by providing a set of principles and practices that gives product and service quality assurance. Top performing organizations ask their DevOps teams to deliver this value, with the right quality, at regular and continuous cadences—in other words, they are implementing continuous quality as part of continuous delivery.

Enterprises, more now than ever before, are in dire need of implementing quality assurance across their systems, engineering, and environments to ensure that quality is being consistently and continuously delivered. As it stands, many companies expect “quality at speed,” yet only check for defects before code is deployed.

QUALITY ASSURANCE & TESTING TOPICS AND DEVOPS ACTIVITIES

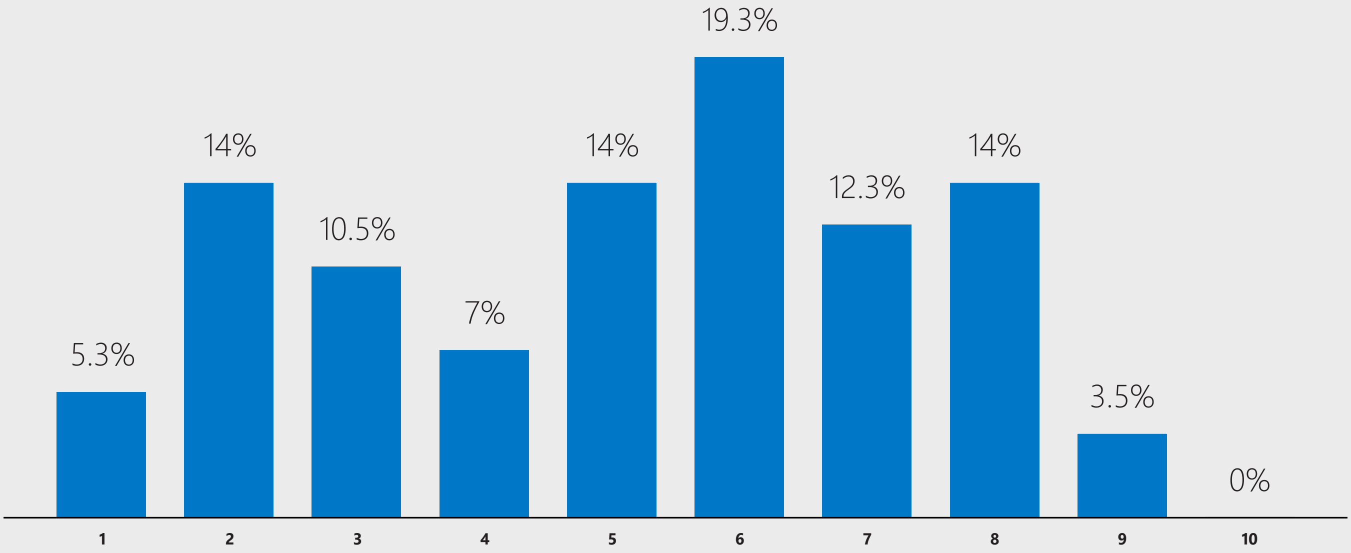
FIG. 6


Quality & Test policy | Responsibility & Roles | Monitoring & Control | Anomaly management | Reporting & Alerting
Quality risk analysis & Test strategy | Acceptance criteria | Quality measures | Reviewing
PLAN
DEPLOY
GRATE
MONITOR
INTE
CODE
OPERATE
Test design | Test data management | Test automation | Test execution | Investigate & Assess outcome
Estimating | Planning | Infrastructure | Tooling | Metrics | Continuous improvement
Organizing topics
Performing topics
DevOps activities

[12] OECD, Definition of Quality- ISO, 2020
[13] Sogeti and Capgemini, Quality for DevOps Teams, 2019

ON A SCALE OF 1-10 HOW WOULD YOU RATE YOUR PROJECT MATURITY ON CLOUD & DEVOPS QUALITY?

FIG. 7



Conversely, leading DevOps organizations are focused on "end-to-end quality". While they are still doing this with thorough code-vetting procedures, they are also able to validate the quality of what is being delivered before, during and after deployment.

Quality should be baked into every process so that the features and characteristics of a product are built right the first time. Quality at an enterprise scale goes far beyond continuous testing and test automation since quality engineering should also focus on delivery, behavior, practices, and many other quality assurance and testing activities. Sogeti Quality Assurance activities framework shows some of the key elements of this approach (Figure 6).

Sogeti's survey amongst cloud and DevOps practitioners revealed an issue with "quality maturity" in the industry (Figure 7). As many organizations have moved from siloed IT teams to having multiple DevOps teams with defined scopes, our surveys show us that the discrepancies between each team's quality practices vary greatly, and there is a need for consistency. When IT professionals were asked to rate their maturity in DevOps and the cloud, their responses showed a wide divergence in maturity levels, as illustrated in the diagram above. From our coordinated research, and in drawing upon some key learnings from Sogeti's book 'Quality for DevOps teams,' we clearly see a number of challenges holding back the maturity of organizations.

What are the challenges?

Balancing speed and quality

Achieving the right balance between quality and speed of delivery has always been tricky, as evidenced by the McKinsey Developer Velocity Index research findings. While half of organizations report that getting a peer review is an important step in the quality process, only 7% of respondents are able to get peer review results in less than a day. And 38% said they have to wait between three and five days, with seemingly trivial delays significantly hindering developer velocity. Enterprises want the right quality at the right moment, and don't want to wait to deliver it. To achieve this balance, many companies have turned to automation.

Test automation is often used as an all-in-one solution to account for the speed/ quality dilemma, but it doesn't solve for the fact that not everything from planning to production can be automated. We need broader and more encompassing solutions.

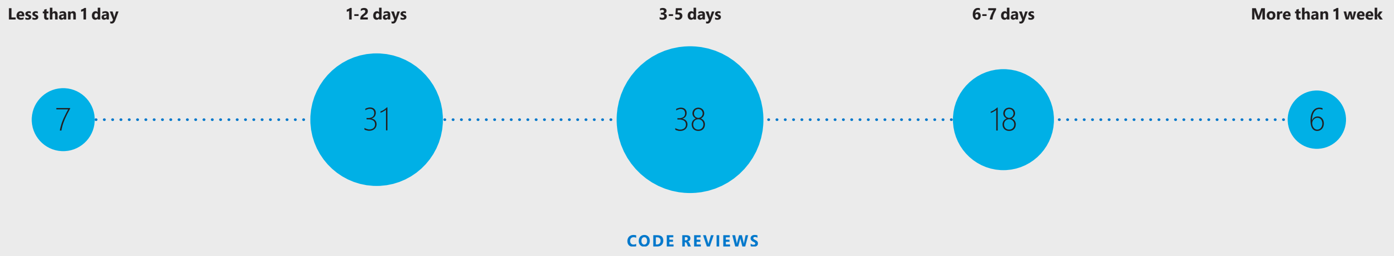
Searching for consistent maturity and enforcing it

Ensuring that cross-functional teams implement quality assurance practices consistently across the enterprise is a key factor in quality management. Successful organizations optimize the efficiency of multiple self-organizing teams by reusing knowledge, while making them adhere to testing frameworks and agreed upon toolchains. Yet, enforcing consistency is an issue, not just in the testing process but across all the design principles, standards and best practices for building DevOps and cloud architectures.

Complex interdependencies

Products, applications, and systems typically have many interdependencies, making the release of new capabilities both challenging and slow. In many organizations, product owners can't simply focus on their own product because a small change (e.g. a regulatory update) may require downstream changes in all the other products, systems and apps that rely on their product. This results in what should be a simple change becoming a complex, multi-team, multi-product update. All of this complexity slows down the release cycle. These interdependencies are not only between DevOps teams, but also with systems and products outside of the influence circle of the organization, like SaaS or COTS, that make it an even greater challenge for DevOps to deliver at the speeds the business requires.

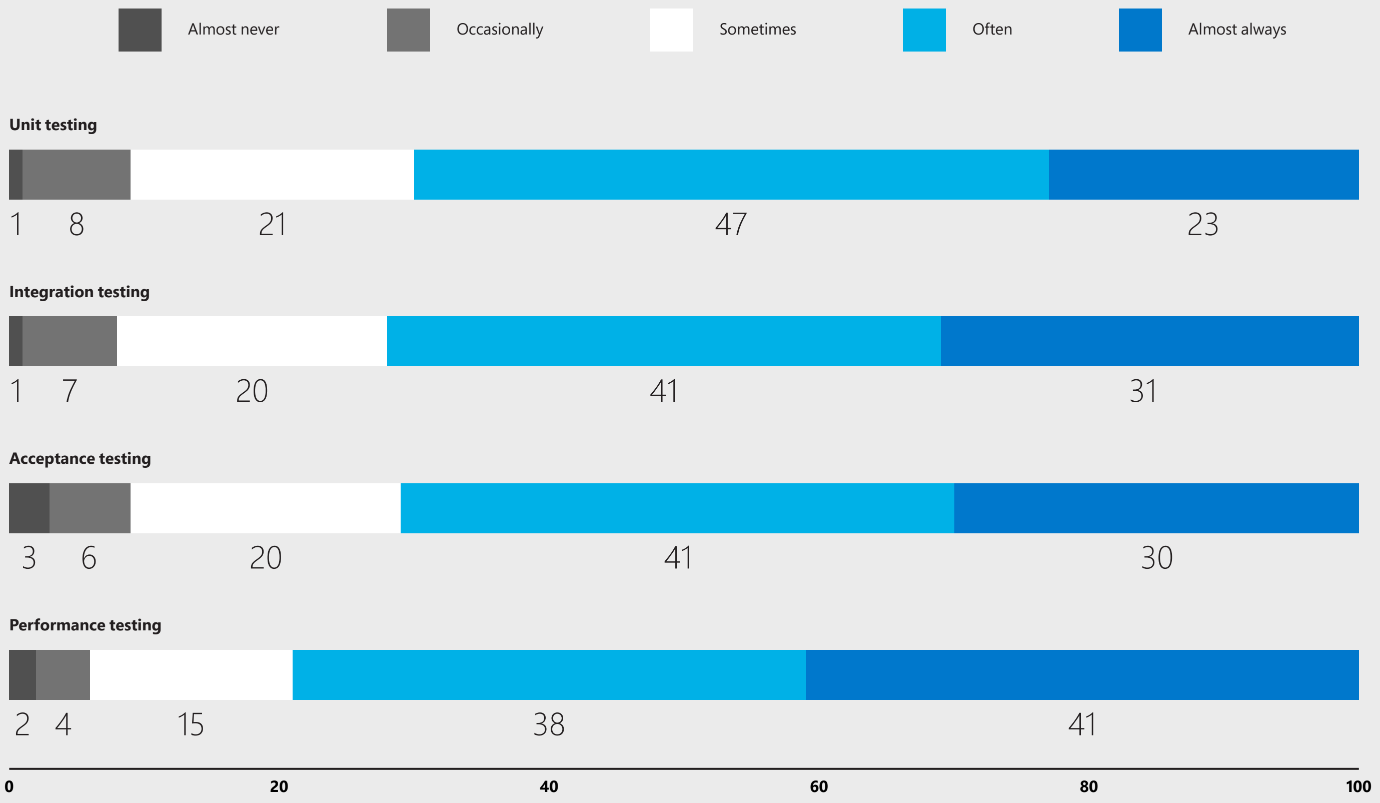
IN GENERAL, HOW LONG DOES IT TAKE FOR DEVELOPERS TO GET BACK THE PEER REVIEW RESULT? *FIG. 8*



AUTOMATED TESTING SYSTEM PROCEDURES

FIG. 9

IN GENERAL, HOW OFTEN ARE THE FOLLOWING TESTING PROCEDURES FULLY AUTOMATED AND INVOLVES MINIMUM MANUAL ACTIVITIES



Lack of test automation

The lack of test automation is certainly not a new challenge and remains a long-term barrier to achieving quality in DevOps. Our research shows that only one third of organizations report “occasionally” or “sometimes” automating their testing processes, meaning many organizations are leaving significant quality gaps and/or relying on slow, inefficient manual testing.

With performance testing virtually impossible to do manually, it comes as no surprise that organizations are twice as likely to automate their performance testing compared to unit testing. While a high level of automation should be a goal for unit testing, a certain level of manual testing isn’t a problem.

Test Driven Development (TDD) drives test automation and gives a continuous assessment of the functional quality of the code. TDD can also lead to major benefits for quality as it forces teams to clarify requirements prior to commencing development. Yet our research shows that 31% of organizations have not consistently adopted a TDD approach. Further, they write unit tests prior to development less than 40% of the time.

Making a cultural shift and defining success

Developers, in general, don't like testing, although good developers do deliver rigorous testing. Enterprises need policies in place that enable developers to deliver quality, without having to be part of a discrete, separate, testing phase. Quality demands a cultural shift, with clear guidelines that testing is the responsibility of everyone, and metrics to measure success. The idea that "what gets measured gets done" applies to many developer teams, and without concrete metrics and insight, teams either won't focus on quality or will try to implement quality checks in the wrong places.

In the past few years, Sogeti interviews revealed that IT teams have seen a steady increase in speed, but also an increase in time to recover from mistakes and implement new solutions—enterprises need to shift their focus from speed back to stability. Quality must be baked into every process, and into the attitudes of developers. Simply knowing the amount of open bugs isn't sufficient to inform teams about the quality of a product developed by the DevOps team. In many instances, the *reporting about* levels of quality is the challenge itself. Instead of simply reporting the number of open bugs, we need to view how testing and systems behave over time. Quality must account for how your DevOps team feels at every stage of the development and release pipeline, from the design through the run stage. If your organization doesn't have good metrics on the system quality, your teams will always be inefficient and your products unstable.

Addressing the challenges

Make the cultural shift with clear quality goals and transparency

If your people know what's required of them, it's easier for them to make the cultural shift towards an end-to-end quality focus in Enterprise DevOps. According to the Sogeti survey, enterprises that are clear and successful in describing the required quality characteristics of systems are much more efficient in test automation, and have higher and more predictable overall system quality.

We know that organizations ascribe to industry standards and regulations, like ISO 25010, for product quality and quality in use, but at a cultural level they must go beyond these external standards. Many top performing enterprises make it clear to their product teams that quality is a communal practice, and by making entire teams accountable, time spent ensuring quality is viewed as a necessity rather than an inconvenience.

Quality needs to be something that enterprises and employees strive for. It must be engrained in the processes and expectations of professional behavior, rather than being viewed as a third-party hoop to jump through.

Understand your software supply chain

90% of IT systems are currently utilizing open-source code, so it is essential for companies to validate the quality of software provided by third-party software supply chains. Without insight into the quality of the supply chains, it's impossible to report fully on the overall quality of a delivered product. Managing the quality of borrowed code inside your software supply chain can be mitigated by using a Software Bill of Material—a document that provides a complete list of all the components in any given piece of software.

Alongside these third-party challenges, the interdependencies also extend into companies' internal DevOps teams. Teams have historically tended to look solely at their own product, taking care of its quality and not thinking about the process of integration and the dependent systems that it will be cascaded into. One way to address this is to put in place specific practices, frameworks, and independent end-to-end regression test teams alongside your deployment strategies to ensure that the overarching system supports each of the small services being deployed.

Determining who takes responsibility for the quality of an entire system, as opposed to a single product, remains an overarching challenge. Top performing organizations often implement a separate team whose sole responsibility is running end-to-end regression tests. These horizontal teams bring automated provisioning of environments and test runs, while taking responsibility for integration of products like COTS applications or SaaS services that don't need the use of a full DevOps team.

Organizations that are more advanced in their DevOps practices should consider tackling the integration of multiple, interdependent products by adopting architectures and a corresponding release strategy specifically designed to support the rollout and integration of new capabilities. These might include canary releases and ring deployments to smaller subsections of customers (beta releases) with highly automated test execution to aggregate feedback without broadly releasing unstable updates.

Ensure consistency with a clear quality policy

Quality needs consistency. And the bridge between the overall business and DevOps teams is vastly improved when principles and policies on quality of delivery are specified at the organizational level. Businesses should be driven by one mission, one vision and one strategy, and DevOps teams must implement these within processes that are embedded in, and supported by, software products. A company's quality policy and test policy translate the mission, vision and business strategy into the principles, approaches and objectives that describe how the organization deals with the people, resources and methods involved in the quality and testing process.

In establishing these overall principles and policies for quality, organizations can pave the way to making it easy for DevOps teams to do the right thing. This will also make team members happier and perform better, and ensure the organization realizes business value with alignment between the teams.

These principles and quality policies should also cover toolchains and test frameworks usage. We know that some organizations give DevOps teams the freedom to choose the tools and frameworks they prefer, but this is only viable if they follow the accepted principles that have been clearly laid out. With too much freedom, teams often become inefficient, which leads to unpredictable quality. What all enterprises need instead are standardized templates for cloud resources and automation designed to deliver consistent enterprise-ready systems.

Balance speed and quality by getting smart with your test automation

While the speed and quality value of test automation is undeniable in the DevOps cycle, many organizations struggle to adopt a system that scales appropriately. The latest Sogeti and Capgemini Continuous Testing Report revealed that just 24% of functional, performance test-cases are automated, while just 22% of test cases are automated within sprints. To combat this, teams must be ready to move from simply automating a checkbox functional test script, and instead position quality and automation at the very heart of the development lifecycle. The key hurdle here is how to remove testing as a bottleneck and make it a lean practice.

A recommendation in the most recent Sogeti and Capgemini World Quality Report[14] is for smart automated frameworks to be designed to support the development cycle. The report adds that smart automation is set to make a significant difference in testing by finding and fixing issues quickly, while helping combined test/development teams decide which changes will deliver the best and fastest returns will be paramount in delivering quality outcomes.

[14] Capgemini, World Quality Report 2019, 2019

QUALITY CASE STUDY

Quality was a key priority for a large transport sector organization migrating from a monolithic legacy system to modern distributed systems. With its existing technology becoming outdated and increasingly difficult to support, the enterprise's goal was to modernize while ensuring the new systems delivered the same (or more) business value as the legacy system.

The new distributed model meant that numerous DevOps teams were working on a range of different systems. As such, the company's senior management sought to maintain a comprehensive overview of the migration progress to ensure that ongoing business would not be impacted by the changes.

Long-term technology services partner Sogeti created a 'confidence monitor' for the business. Using tools like an automated real-time dashboard, all teams (10+ in number, mostly DevOps and Scrum) regularly published their information on the quality and risks of the new systems.

Based on this information, Sogeti analyzed a range of measured indicators, including both hard data (such as information about successful installations, functionality, performance, and business continuity) and soft data (such as a periodic questionnaire to the operations team and end-users). The data was compiled into a single "confidence rating." This showed how confident the migration delivery teams were in the robustness and quality of the different systems. It also worked to identify areas where more work was needed, and showed which systems were working as planned.

Every month over the course of several years this confidence rating was reported to the company's senior management, and became a reliable compass to steer the organization's overall transition to a modern IT landscape.

QUALITY NEXT STEP RESOURCES

Make the cultural shift with clear quality goals and transparency
Read more on the Continuous Testing Report

Understand how your supply chain and DevOps teams interconnect
Learn how GitHub secures open source software

Ensure consistency with clear quality policy
How Sogeti defines clear policies in quality

Balance speed and quality by getting smart with your test automation
Read more in the Capgemini World Quality Report

Security

Enterprise DevOps and, with it, widescale cloud adoption, represent a seismic shift for information security teams.

Information security experts have access to capabilities they could only previously dream about. For example, we now have the ability to trace any piece of code from production deployment backward through the software development lifecycle, understand the validations that have taken place at each stage, all the way back to the initial code check-in. However, we also now live in a world where many teams are pushing multiple code deployments per day, each of which has the potential to introduce an exploitable vulnerability if security principles are not embedded across the entire DevOps lifecycle and organization.

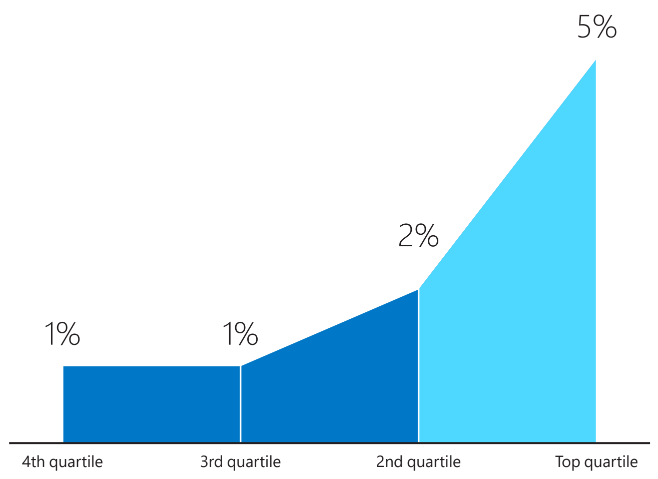
Of course, information security is a wider discipline than just the application code-focused example cited above. As the ISO 27002 (code of practice for information security controls) standard states; *"... information security management requires, at a minimum, participation by all employees in the organization"*. Security also has a real impact on the business—as the DVI research clearly showed that security and compliance was the second highest weighted driver (23%) on the difference between top quartile and second quartile enterprise performance. Security isn't just a cost, it's also a driver of performance.

DevOps helps organizations release software faster, but it can also help address the security needs when both developing software solutions and then running it in production. When security is embedded throughout an entire organization, it helps drive faster and more secure software delivery and assists in achieving consistent governance and control. Similarly to governance, when teams feel responsible for their products, and security is a communal goal, products are much more likely to be secure, and DevOps contributors will embed it into their systems and code. Security must be embedded in the full lifecycle of systems and teams, and top performing companies are turning to automation more frequently to achieve this consistency in a cost-effective way.

Unfortunately, there is no "one size fits all" approach to security. Rather, each application has its own unique requirements and constraints, so practices need to be tailored to its processes, security requirements, and operational procedures. What enterprises need are a set of principles and regulations that guide their security implementation across the complete software development lifecycle (SDLC).

What are the challenges?

REVENUE CAGR, 2014-2018 *FIG. 11*



Lack of a comprehensive approach to security

17% of organizations surveyed for the DVI report said they only test security vulnerabilities "for a major release" or "only when releasing to production", and only 46% of enterprises are incorporating security tooling into their DevOps pipelines. While 68% of respondents said that "security was everyone's responsibility" only around 40% were incorporating security requirements in the design phase, collaborating on threat models and prioritizing security requirements as part of their development backlogs.

Taken collectively, it's clear that organizations are not taking the holistic, systems thinking approach to security needed to implement a high-speed DevOps model securely. This is reinforced by the survey results regarding response time to resolve major security breaches. 6% of respondents said it would take "weeks", with 17% saying "between 3 days to a week", to resolve. Only 7% said they had the ability to resolve a major breach in less than 1 hour.

Securing open source

One of the other fascinating insights to emerge from the DVI research was the gap between the adoption of open source and the ability for companies to use their open source code securely. Despite 65% of organizations saying they "often or almost always leverage open source for product development", less than 20% are automatically scanning those components for vulnerabilities and remediating those vulnerabilities before deployment.

Deploying code with known vulnerabilities leaves organizations vulnerable to attack. Recent research in the State of Open Source Security Vulnerabilities report showed the number of OSS vulnerabilities jumped from 4,100 in 2018 to 6,100 in 2019, a 50% year-over-year increase.

WHICH OF THE FOLLOWING SECURITY PRACTICES CURRENTLY APPLY TO YOUR ORGANIZATION? *FIG. 10*

68%
Security is everyone's responsibility (e.g., there is a centralized security team; skilled security experton almost every team;
every developer is responsible for securing his/her code

47%
We have a centralized team to manage the security of open source code and package used by our organization

46%
Security tools are integrated in the development integration pipeline

44%
Security personnel review and approve code change before deployment

34%
We have a centralized team of security researchers

42%
Security and development teams collaborate on threat models

41%
Security requirements are treated as design constraints and product owners engage with security experts as early as the requirements definition and design stage

40%
Security requirements are prioritized as part of the product backlog

30%
Developers can provision a security of open source code and package used by our organization

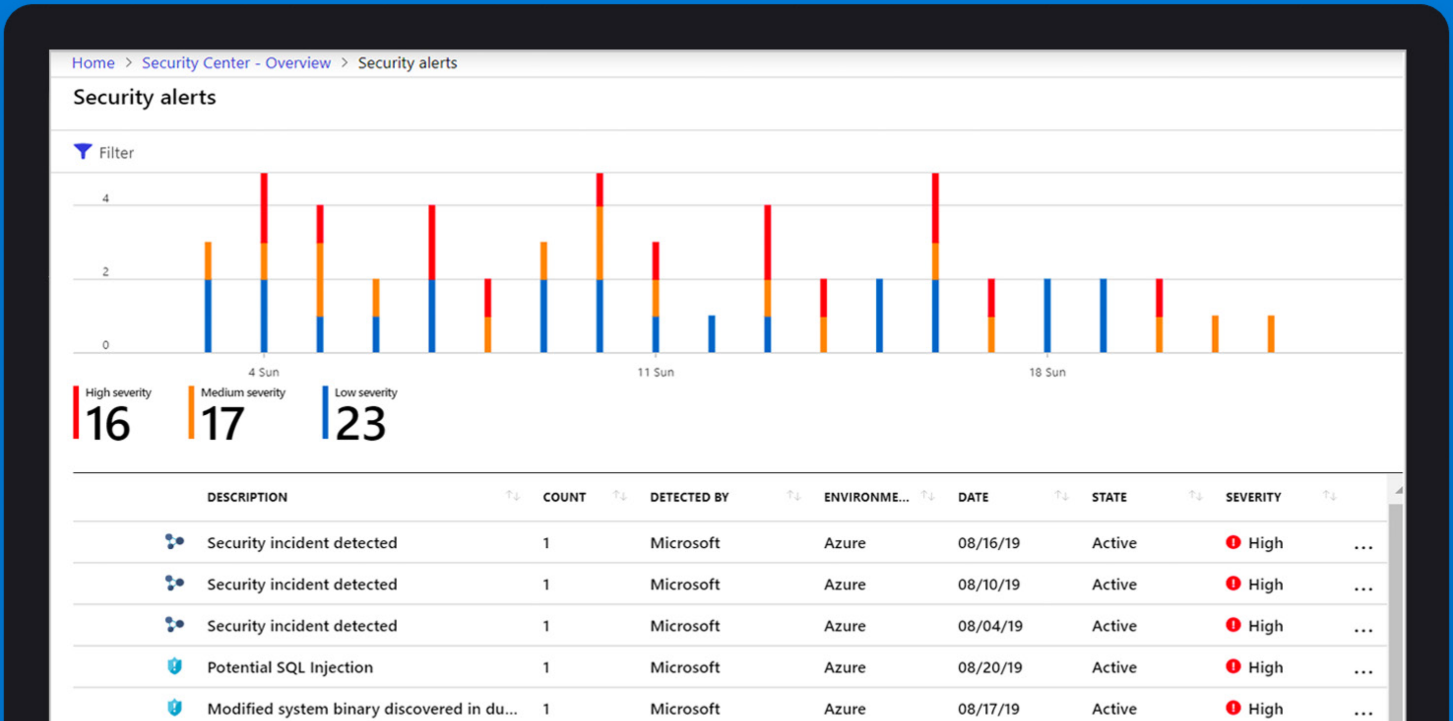
While open-source adoption offers huge benefits to enterprise organizations, it also raises major security challenges. Organizations need to understand where open source is being used within their applications and consistently adhere strictly to the open source security best practices across the software development lifecycle.

The cloud security challenge

Despite cloud vendors' best efforts, many enterprises seem to be struggling with securing their cloud environments. There have been numerous high profile and well-publicized cases of security breaches in cloud environments, through human error resulting in misconfigured systems, lack of key skills, negligence, or company insiders with malicious intent. Many companies are struggling with the transition from the traditional data center "perimeter (network) security" model to the multi-layered "defense-in-depth" approach recommended in cloud environments. Security professionals are having to learn new skills and new approaches in order to deliver effective cloud security, and for many organizations this transformation is lagging behind their cloud adoption, creating a security gap that leaves them vulnerable.

Lack of board-level oversight and investment

While an increase in federal and multi-national data regulations has led to a big uptick in enterprise data security initiatives, major hurdles have included the lack of organizational focus on security and inadequate funding for security initiatives. Nearly half of organizations have cybersecurity on their board agenda at least quarterly, according to Deloitte's Future of Cyber Survey[15]. However, only 4% of respondents say cybersecurity is on the board agenda once a month, despite increasing regulatory pressures that make the company directors personally liable for data security.



	DESCRIPTION	COUNT	DETECTED BY	ENVIRONME...	DATE	STATE	SEVERITY	
	Security incident detected	1	Microsoft	Azure	08/16/19	Active	High	...
	Security incident detected	1	Microsoft	Azure	08/10/19	Active	High	...
	Security incident detected	1	Microsoft	Azure	08/04/19	Active	High	...
	Potential SQL Injection	1	Microsoft	Azure	08/20/19	Active	High	...
	Modified system binary discovered in du...	1	Microsoft	Azure	08/17/19	Active	High	...

Addressing the challenges

Adopting DevOps, with its focus on automation and shared responsibility, can result in secure systems that are compliant with security guidelines and able to withstand the security threats that exist, if implemented properly.

Addressing the challenges of security as part of an Enterprise DevOps transformation requires a combination of key principles, standardized architectures (aka "platforms"), organizational design and automation.

Adopt the Zero Trust model

The fundamental mindset shift, particularly for enterprise organizations that have moved from on-premise hosting to a cloud-first model as part of their DevOps adoption, is the Zero Trust model. Microsoft defines Zero Trust as: "Instead of assuming everything behind the corporate firewall is safe, the Zero Trust model assumes breach and verifies each request as though it originates from an open network… Zero Trust teaches us to "never trust, always verify."

Guiding principles of Zero Trust are:

- **Verify explicitly.** Always authenticate and authorize based on all available data points, including user identity, location, device health, service or workload, data classification, and anomalies.
- **Use "least privileged access."** Limit user access with Just-In-Time and Just-Enough Access (JIT/JEA), risk-based adaptive polices, and data protection to protect both data and productivity.
- **Assume breach.** Minimize the blast radius for breaches and prevent lateral movement by segmenting access by network, user, devices, and application awareness. Verify that all sessions are encrypted end-to-end. Use analytics to get visibility, drive threat detection, and improve defenses.

To read more about how your enterprise can adopt a Zero Trust model, click here.

Secure development lifecycle and shift left

Ever since Bill Gate's now famous Trustworthy Computing memo in 2002, the role of software developers in security, and the importance of embedding security within the software development lifecycle[16], has become part of accepted industry best practice.

Enterprises must build security best practices into the way they design, write, build and test applications to dramatically reduce the number of exploitable vulnerabilities that make it into production. Setting clear security standards and building them into both the day-to-day development and operational practices, while embedding them into the platforms used by teams, is the first step. This will remove the over-reliance on centralized security teams that are so often overwhelmed with demands caused by the rapid rate of DevOps change.

Sogeti research shows that top performing enterprises introduce security checks at all phases of the software lifecycle—plan, develop, build, test, release, deliver, deploy, operate and monitor. Application security must speed up to keep pace with operations by bolting security checks and control inside the CI/CD pipeline itself. Testing and security should be "shifted to the left" through the automated unit, functional, integration, and security testing. This is a key differentiator since security and functional capabilities are tested and built simultaneously.

Secure reusable code components

Creating secure reusable code components, and making them widely used via your InnerSource strategy, is also key in solving Enterprise DevOps security.

Top performers are consistently using reusable components and leveraging what is known as the "DRY principle"[17] (Don't Repeat Yourself). DRY has the added benefit that you only need to fix a security vulnerability identified in your reusable code module once to fix that vulnerability for everyone, compared to potentially hundreds of times if the code has been reused via 'copy & paste'.

Obviously, open source software is a key source of reusable components for many enterprise organizations and improving the security of these components is an important industry goal. Top performers are installing automated security remediation technologies, like GitHub's Dependabot[18], which ensures security and visibility across the open source software supply chain. Dependabot pulls down your third-party open source code and checks for any outdated or insecure requirements. It then automatically flags what sections are out of date and allows you to make required updates to ensure that all code deployed is secure and compliant.

Build security into reference architectures and platforms

Platforms & executable reference architectures are a great place to start when dealing with Enterprise DevOps security. Central security teams help to define a set of secure reference architectures, ideally defined as-code and these are then shared using the InnerSource model discussed earlier. Solid examples of this practice can be found in the Azure Architecture center. These secure reference architectures, ideally along with secure release pipelines, provide a path of least resistance, so that doing things the *right* way securely, also becomes the *easiest* way.

Successful organizations are adopting DevOps reference architectures, a template documenting a system's full set of activities, steps, practices, and technologies that are used when creating an application or technology. With all the ideal steps, from build to monitoring, laid out clearly, enterprises can be confident that things are being done the "right" way, whilst addressing all of the organization's security goals at the same time.

When using these organizational reference architectures, teams often then create DevOps Blueprints, which extract a team's exact needs along each stage of the system's reference architecture. By figuring out exactly where along the software development lifecycle teams will need to use different technologies, expectations are clear. This ensures that DevOps teams know exactly what they must do to achieve business and security goals.

Observability, SIEM and SOAR

In a traditional cybersecurity world, Security, Information, and Event Management (SIEM) tools provide a unified security analytics platform to help the enterprise identify threats and malicious activity across the entire organization. Increasingly, top performing companies have started leveraging artificial intelligence (AI) to make sense of the terabytes of data that are collected by security monitoring tools. AI can help to provide deeper insight into what is happening within the Secure Development Lifecycle and to determine the best response to the security threats.

[16] Microsoft, SDL Practices, 2020
[17] Microsoft, Patterns and practices for Super DRY development for ASP.NET Core, 2019
[18] Dependabot

But, in order to aggregate this security and performance data, the applications and systems needs to be "observable". This means systems must be designed to expose important operational data via logging or other interfaces so that enterprises can "see inside" to understand how any given system is performing. The requirements for observability need to be part of the application design, and clear parameters must be set regarding what data is needed to understand the health of an application. This is vitally important because good observability enables us to baseline the behavior of an application under normal conditions. Anomaly detection can then be used to spot abnormal behavior that might be the result of malicious activity.

Having near real-time insight into the health of applications and infrastructures, and the security threats they face, enables DevOps automation techniques to be leveraged to automate some of the responses to these attacks. Some organizations are using the acronym SOAR (Security Orchestration, Automation and Response) to describe this process. Applications like Azure Sentinel can trigger an automated response when an alert threshold is met. These automated playbooks can deal with common attacks or issues without needing human intervention.

Make security a board-level responsibility

Security is too often seen as an IT problem, but building up a risk-aware culture should start at board level. Failing to invest in security, and not ensuring that critical security safeguards are in place, leaves companies open to liability. From our interviews, it is clear that enterprises must introduce security as a monthly checklist item at board meetings, and define a rubric of compliance that helps them adhere to the set of decided upon policies.

The National Cyber Security Center[19] offers a great list of questions that executive boards and CEOs should constantly be asking themselves. Most importantly, does your enterprise have a full and accurate understanding of:

- The impact on the company's reputation, share price or existence if sensitive internal or customer information held by the company were to be lost or stolen?
- The impact on the business if online services were disrupted for a short or sustained period?

Boards and executive-level leadership must always think large picture, and while largely "unseen" by customers, security concerns must always be a top priority to avoid costly security failures.

[19] National Cyber Security Centre, Introduction to Cyber Security- Board Level Responsibility, 2019

SECURITY CASE STUDY

Security is high on the agenda for Dutch electricity and gas provider Enexis. As a key player in the Netherlands' energy infrastructure, its public profile states: "Our 4,500 or so dedicated employees are working hard every day to ensure stable and reliable grids and to secure the future of our energy supply."

A scalable and agile IT-capacity in a public and cloud-native environment is key to this, and the company worked with Sogeti to bring about the transformations needed to succeed. In cooperation with partners, Sogeti created a roadmap and transitioned more than 100 existing applications to a cloud-native application platform, either on IaaS, PaaS or SaaS.

Security sat at the core of this process, with part of the success being the risk classification of data and systems that moved to the cloud. Within a reference architecture, Sogeti defined the risk levels for building capabilities in the cloud, with principles for both security and cloud.

With measures appropriate to acceptable levels of risk, the Enexis Platform DevOps teams delivered standard services (e.g. network, servers, etc.) with embedded security, and automatic patching. For both the Platform DevOps and the Apps DevOps teams using these standard services, the security policy based on this risk classification was translated into concrete requirements for the cloud platform. In this way, the teams could easily carry out a 'cloud readiness assessment review' before deploying to production.

This secure Enterprise DevOps model was further enabled by the simplification of applications implementation and integration within the IT operation using Continuous Integration and Continuous Deployment (CI/CD). The results were impressive. Application delivery went down from 6-12 weeks to less than 2 days, with the shift left approach enabling security issues to be identified and fixed early in the cycle to reduce overall lifecycle costs.

SECURITY NEXT STEP RESOURCES

Adopt the zero trust model
Learn more about the Zero Trust model
See how Microsoft assesses Zero Trust policies

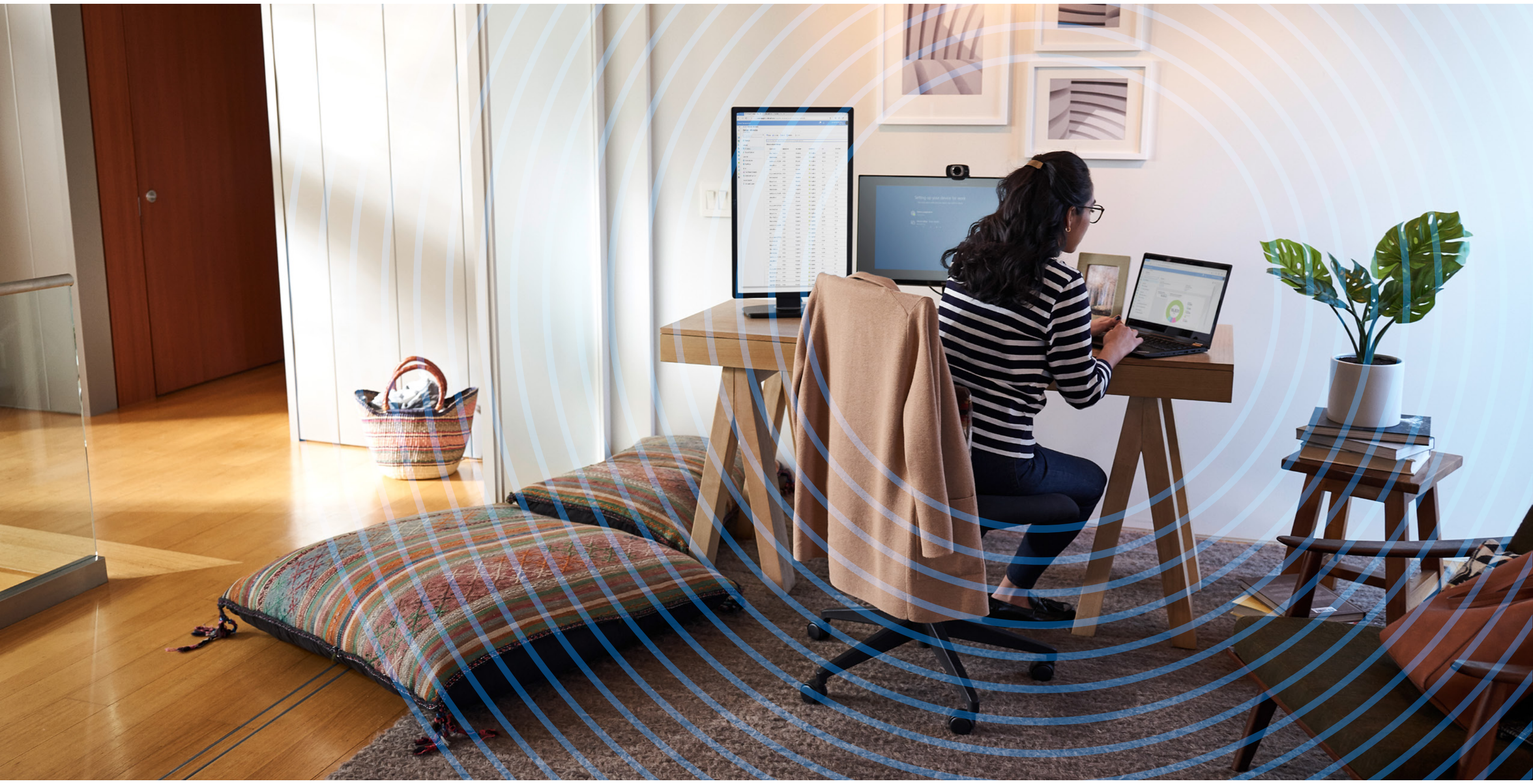
Secure development lifecycle and Shift Left
Learn how to bake security into your development lifecycle

Secure reusable code components
Read more about the DRY principle
See how GitHub implements Security

Build Security into reference architectures and platforms
Learn more about the Azure Architecture center

Observability, SIEM, SOAR
See Microsoft's Azure Sentinel
Read the Intro to Observability
Automate security responses with playbooks

Compliance



Compliance poses many of the same challenges as governance, security, and quality for Enterprise DevOps.

However, while those areas are mainly determined within an organization, compliance is driven largely by external regulation, and companies unable to prove that they are operating within the rules may face significant legal or financial consequences.

Enterprises always strive to be compliant with regulations, but as more applications emerge in the technology landscape and offer similar services as existing companies, compliance accreditations are hugely important for the end customers in choosing which services they will use. Customers, as well as the enterprises themselves, want to be confident that their information and data is secure, safe, and uncompromised.

Navigating the complex web of local, global, and industry-specific regulations in a constantly moving DevOps environment is challenging for many organizations. These challenges fall into two categories: first, building, automating, and running compliant systems; and second, proving system compliance to regulators.

What are the challenges?

Compliant with what?

Almost half of executives surveyed by Microsoft said they were not sure which data compliance standards they needed to meet. This number is likely to increase as enterprise organizations embrace new business models as part of their digital transformations. For example, as more businesses begin to take online payments, they will fall under the scope of the Payment Card Industry Data Security Standard (PCI DSS) for the first time. Other industries that face significant data security regulation include retail (through the GDPR in the EU) and healthcare (HIPPA), among others.

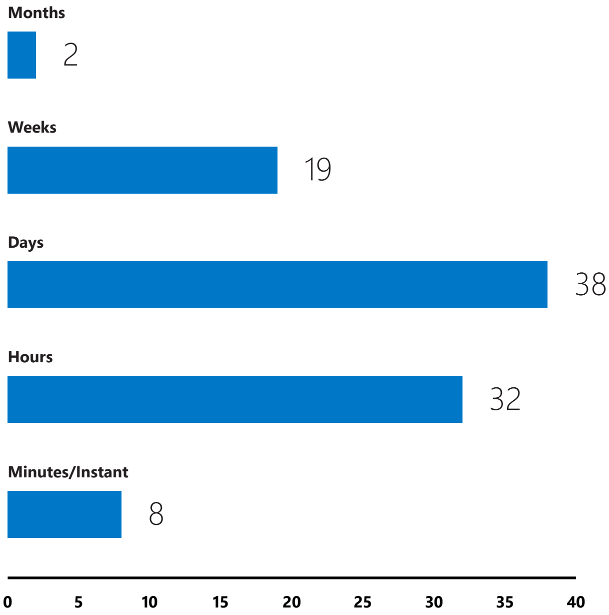
The GDPR & data security challenge

The EU's General Data Protection Regulation (GDPR), which came into effect during May of 2018, has caused major compliance challenges for enterprises globally. GDPR encompasses any organization that processes EU citizen data, regardless of where that company is incorporated, or where the data is processed or stored. GDPR has forced many senior executives to focus their attention on governance because of the potential for huge fines under the new legislation.

However, as organizations embrace DevOps, it is very common to see data replicated away from traditional centralized 'systems of record' into distributed data stores. These data stores might also be moved from on-premise to cloud-hosted environments or use new database technologies that existing governance processes aren't equipped to govern effectively. This increased architectural complexity can create challenges for organizations attempting to comply with data security regulations.

IN GENERAL, HOW LONG DOES IT TAKE FOR REGULATORY COMPLIANCE CHECKING?

FIG. 12



The speed of compliance

Microsoft's research, as part of the DVI survey, showed that 59% of respondents said it can take "days or weeks" to assess their current state of regulatory compliance. In DevOps, where multiple releases of new software features might happen every day into constantly evolving cloud environments, the feedback cycle to understand your current state of compliance needs to be much faster. Simply verifying that an application or environment is secure when it is first deployed as part of its initial release is no longer sufficient. The problem of 'configuration drift', where the application and its underlying infrastructure 'drifts away' from the originally defined compliant state as the result of small incremental changes over time, poses a major challenge, and threat, to enterprises.

Addressing the challenges

As previously stated, in order to address the compliance challenges within Enterprise DevOps transformation, enterprises need to achieve two goals. Firstly, the organization must *be compliant,* and secondly, it needs to be able to *demonstrate compliance* to a regulatory body. DevOps, if implemented and scaled correctly within organizations, can help solve both these requirements, at a far lower total cost of ownership than previous operating models.

The compliant platform

Much like security, compliance can be 'built-in' to reference architectures and shared platforms so that as much of the burden of compliance can be abstracted away from product-aligned DevOps teams. For example, Microsoft publishes reference architectures for common compliance standards, much like they would for security. These architectures go beyond simple Visio diagrams and Word documents to include the DevOps-automation steps required to fully build a compliant environment in the Azure cloud.

Organizations are designing and building 'compliant' platforms based on these reference architectures, which are often maintained with InnerSource practices. These shared compliance platforms can be either centrally managed as a landing zone for applications built by DevOps teams, or instantiated on-demand by DevOps teams, secure in the knowledge they meet the compliance standards.

Continuous compliance-as-code

Enterprise DevOps can help address the "configuration drift" challenge with the concept of continuous compliance. DevOps automation methods like infrastructure-as-code and configuration-as-code create a code model of how we expect the underlying compute environment to look. Similarly, CI/CD pipelines, with embedded test automation, define how applications are built, deployed and how they should behave. When combined, they give the ability to continuously apply compliance standards by comparing the model (expressed as code) with the reality as running in production, and automatically correct any drift.

The key to this is a concept called 'idempotence'. Idempotence, in the language of DevOps, refers to the ability to redeploy the same desired state configuration or template to the existing environment, regardless of the current state of that environment. Idempotent configuration templates enable enterprises to use 'declarative syntax' to describe how they want their environment to look. In other words, they 'declare' they want it to be a certain way, and idempotence means that, whether it's the first time or the hundredth time this 'declaration' has been issued, regardless of its current state, the tools will make it so. Idempotence is particularly important in automation, as an enterprise can say "every half hour, make sure an environment looks like this...". Then every time that declarative syntax is approved, you know when you run a command, everything will be validated, nothing will be duplicated, and your environments will remain continuously compliant.

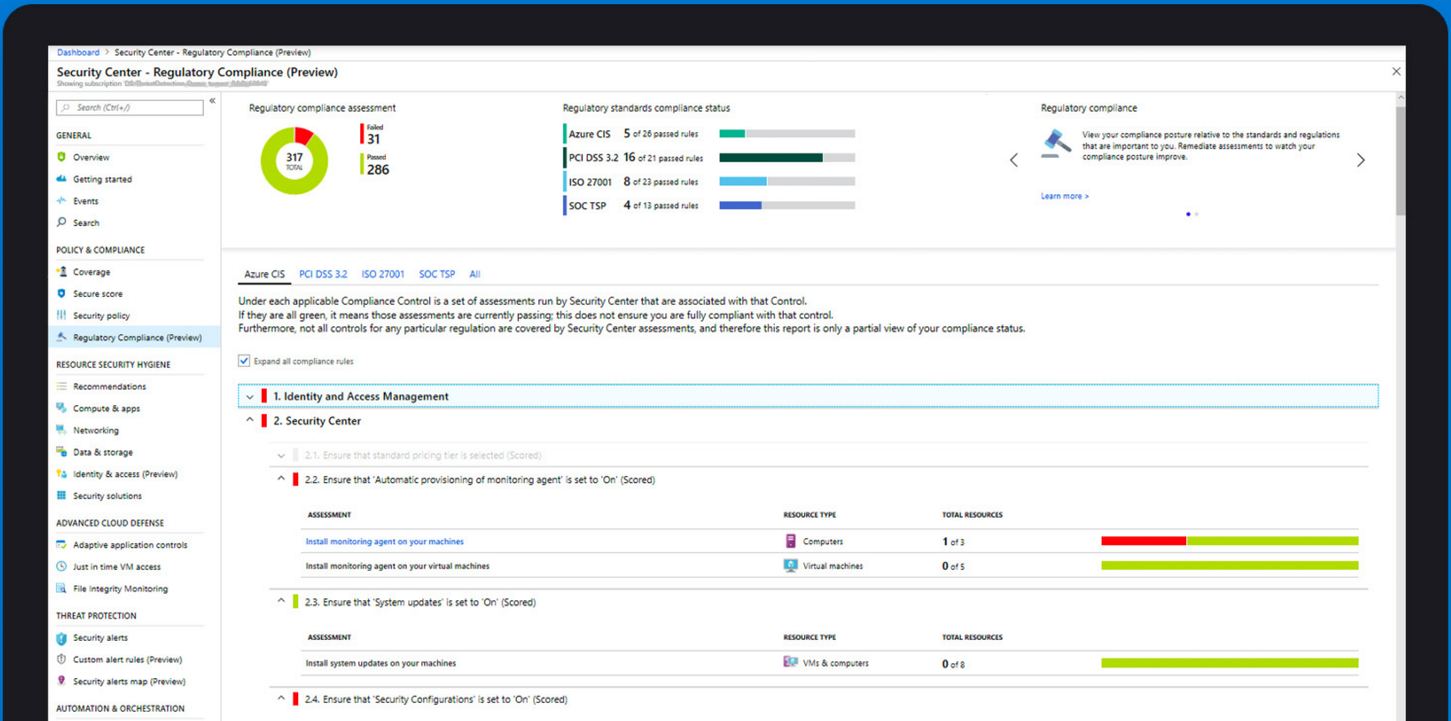
Compliance monitoring and reporting

Being compliant isn't enough, however—you need to be able to report on and document your compliance status to meet the needs of both internal compliance teams and external audits from third parties or regulatory bodies.

DevOps-enabled teams that have embraced the "as-code" and "continuous compliance" models have an inherent advantage. The as-code configurations form part of the documentation of your compliance framework, and the automated reporting from your continuous compliance process demonstrates the application and enforcement of said framework. More and more frequently, tech-savvy third-party auditors are assessing where organizations have leveraged industry-certified as-code templates as part of their compliance framework, and companies who can quickly demonstrate consistent compliance have a smoother audit experience.

Top performing enterprises are implementing compliance dashboards, similar to the application monitoring solutions that operations teams have used for years. The goal of the compliance dashboard is to give both the DevOps teams and centralized compliance teams real-time intelligence into their compliance status. For example, the Azure Security Center has a Regulatory Compliance dashboard as shown in Figure 13.

These dashboards simplify the process of measuring and reporting on compliance, but still provide the ability to drill down into granular details if necessary.


Dashboard > Security Center - Regulatory Compliance (Preview)
Search (Ctrl+/)
GENERAL
Overview
Getting started
Events
Search
POLICY & COMPLIANCE
Coverage
Secure score
Security policy
Regulatory Compliance (Preview)
RESOURCE SECURITY HYGIENE
Recommendations
Compute & apps
Networking
Data & storage
Identity & access (Preview)
Security solutions
ADVANCED CLOUD DEFENSE
Adaptive application controls
Just in time VM access
File Integrity Monitoring
THREAT PROTECTION
Security alerts
Custom alert rules (Preview)
Security alerts map (Preview)
AUTOMATION & ORCHESTRATION
Regulatory compliance assessment
317
TOTAL
Failed
31
Passed
286
Regulatory standards compliance status
Learn more >
Azure CIS
PCI DSS 3.2
ISO 27001
SOC TSP
All

Security Center - Regulatory Compliance (Preview)

Showing subscription '[redacted]'

Regulatory standards compliance status	
Azure CIS	5 of 26 passed rules
PCI DSS 3.2	16 of 21 passed rules
ISO 27001	8 of 23 passed rules
SOC TSP	4 of 13 passed rules

Regulatory compliance

View your compliance posture relative to the standards and regulations that are important to you. Remediate assessments to watch your compliance posture improve.

Under each applicable Compliance Control is a set of assessments run by Security Center that are associated with that Control.
If they are all green, it means those assessments are currently passing; this does not ensure you are fully compliant with that control.
Furthermore, not all controls for any particular regulation are covered by Security Center assessments, and therefore this report is only a partial view of your compliance status.

☑ Expand all compliance rules

1. Identity and Access Management

2. Security Center

2.1. Ensure that standard pricing tier is selected (Scored)

2.2. Ensure that 'Automatic provisioning of monitoring agent' is set to 'On' (Scored)

ASSESSMENT	RESOURCE TYPE	TOTAL RESOURCES
Install monitoring agent on your machines	Computers	1 of 3
Install monitoring agent on your virtual machines	Virtual machines	0 of 5

2.3. Ensure that 'System updates' is set to 'On' (Scored)

ASSESSMENT	RESOURCE TYPE	TOTAL RESOURCES
Install system updates on your machines	VMs & computers	0 of 8

2.4. Ensure that 'Security Configurations' is set to 'On' (Scored)

COMPLIANCE CASE STUDY

Utility company Eneco had bold ambitions to stay ahead in innovation, sustainability and customer centricity, while becoming more international. It recognized that its existing locally-based IT services would need to change to enable this. The organization decided to move to a centralized group-wide platform for IT.

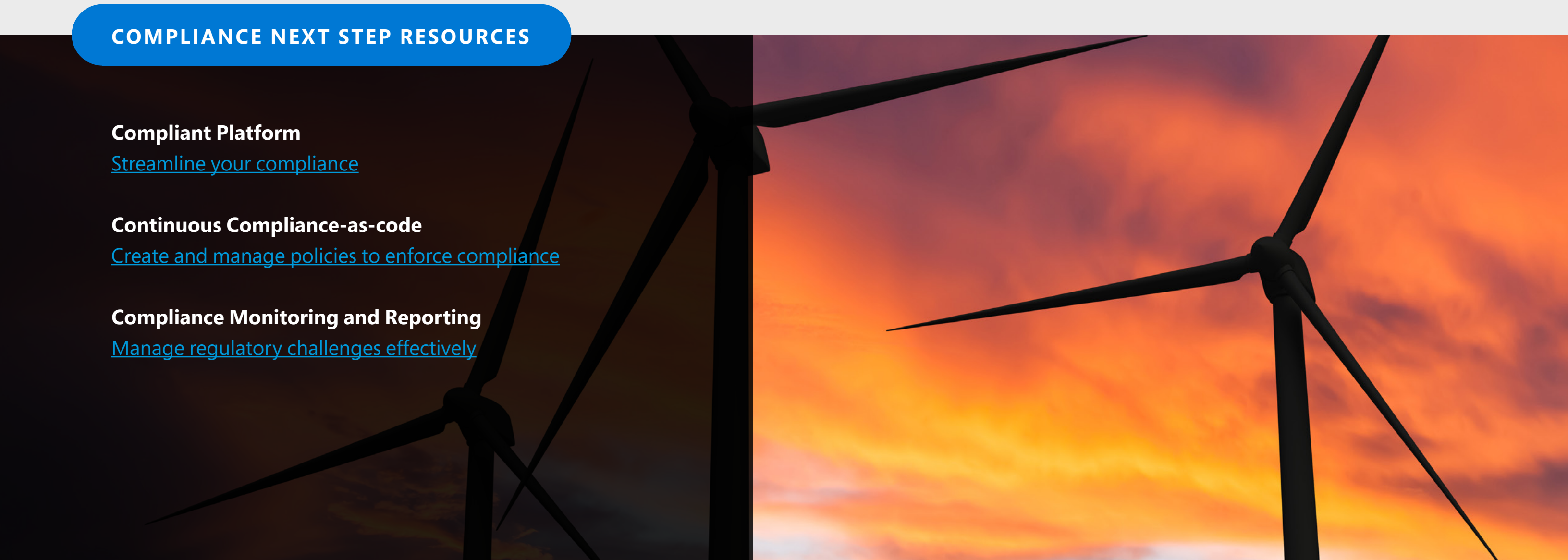
But how could Eneco ensure continuous control of this new, globalized technology landscape, particularly within the cloud-based IT services and DevOps teams? The company still operated with some on-premises legacy technology but had plans to become wholly cloud-based by 2022. With more and more people moving into new cloud-based roles, there needed to be a level of consistency across the entire organization.

Working with Sogeti, Eneco established a cloud reference architecture that described the cloud, business and security principles and standards by which the IT landscape should be run. Compliance with this reference architecture would ensure that Eneco could actively manage risks, information (flows), compliance, costs, and quality. That same reference architecture played a major role in alleviating external compliance concerns by governing the use and security of data and applications in the DevOps process, as well as partner-hosted applications and services.

As the reference architecture demanded innovative but new ways of working, it was important to enforce its use for teams unfamiliar with the principles and standards. Today, DevOps product owners have more autonomy and ownership in cross-functional teams but still operate within Eneco's reference architecture framework.

Furthermore, a Cloud Foundation with a landing zone built on Microsoft Azure provides all the generic services (including all basic and advanced security services) under a single managed services DevOps team—in compliance with the reference architecture. This has freed up all other DevOps teams to focus on creating new capabilities, responding to immediate events, and scaling up (or down) virtual machines within minutes. They can now meet unexpected business need without having to worry about whether the VMs are up to date with all compliance needs as required by the defined principles.

COMPLIANCE NEXT STEP RESOURCES



Compliant Platform
Streamline your compliance

Continuous Compliance-as-code
Create and manage policies to enforce compliance

Compliance Monitoring and Reporting
Manage regulatory challenges effectively

Conclusion

Our research has shown that Developer Velocity, as measured by the Developer Velocity Index (DVI), is strongly correlated with improved returns to shareholders, as well as other indicators of business performance like innovation, customer satisfaction, brand and talent management.

Enterprise organizations are embracing DevOps as they seek to both improve Developer Velocity and to enable a wider digital transformation. As these digital transformations continue, IT departments are undergoing a DevOps transformation to ensure they can meet the evolving needs of the organization and its customers in a rapidly shifting digital economy.

It is clear that enterprise-scale DevOps transformation can create challenges, particularly in relation to product management, distributed and remote working, governance, quality, security, and compliance.

However, when taken holistically, we see top performing organizations following common ideologies and patterns to address these challenges.

TOP PERFORMING ENTERPRISE ORGANIZATIONS ARE:

1. Shifting to autonomous, product-aligned teams that use continuous delivery practices, DevOps toolchains, and cloud hosting to **accelerate product delivery and improve developer velocity.**

2. Implementing shared, self-service platforms that have the necessary governance, compliance, security and quality built-in, **making the ‘right’ thing to do also the ‘easy’ thing to do.**

3. Embracing an InnerSource mentality to share and re-use code and best practices across the enterprise, but also to **foster a collaborative ethos** of continuous improvement with shared ownership and accountability.

4. Leveraging automation wherever possible to reduce the time spent on repetitive work, freeing teams to **focus on providing business value and innovating,** while avoiding operator overload when dealing with massive volumes of data, alerts, etc.

5. Devising standardized reference architectures around common cloud infrastructure and application patterns, then using DevOps automation tactics like infrastructure-as-code, and configuration-as-code to move these reference architectures from Visio diagrams into **reusable code templates** that accelerate widespread adoption.

6. Breaking down the silos, not only between Dev and Ops, but also between the centralized teams that establish the standards of quality, security, and governance, and the product teams that than need to conform to these standards. Teams that have contributed to the standards and governance processes are **more likely to conform to them.**

7. Measuring and learning—observability is key to the success of enterprises. It not only enables them to measure quality and performance, but also to report on it, to thus create the feedback loops that enable organizations **to learn, improve and optimize all their processes and technology.**

The recommendations in this report can be used as a blueprint to address the challenges highlighted and to accelerate DevOps transformation. This, in turn, will enable those transforming enterprise organizations to achieve both better customer outcomes and better returns for stakeholders.

Authors

Samit Jhaveri is the Director of Product Marketing with Microsoft Azure focused on cloud application development and DevOps with GitHub. He serves as the business leader working across product management, sales leadership & finance with responsibility for defining and executing the e2e go-to-market strategy including pricing & offers and execution plans such as campaigns and field & partner motions for growing the business. He is also accountable for Azure value proposition for app development and testing including positioning and messaging, content development and branding. Prior to the current role, Samit led an engineering team at Microsoft's Server and Tool Division and was responsible for shipping several B2B solutions for different vertical industries. Samit earned his MBA from the University of Washington and Masters in Management Information Systems from the University of Arizona.

Clemens Reijnen is Sogeti's Global CTO Cloud Services and DevOps leader. He has been awarded the Microsoft Most Professional Award for 10 years in a row and is a SogetiLabs Technical Fellow. He co-authored the book 'Collaboration in the Cloud' with Microsoft and writes frequently on cloud and DevOps on Sogeti.com. As a global DevOps leader, he works closely with Sogeti's large enterprise customers to ensure their cloud adoption and Enterprise DevOps transformation programs make value for the business.

Steve Thair is the CTO and co-founder of DevOpsGroup Ltd, a leading UK cloud & DevOps services company. Founded in 2013 DevOpsGroup makes DevOps and Cloud adoption fast, secure and simple, helping organizations thrive in the new Digital Economy. In recognition of his DevOps thought leadership and community contributions in co-founding www.winops.org, a community dedicated to fostering DevOps patterns and practices in Windows and Microsoft technology environments, in 2016 Steve Thair joined Microsoft's "Regional Director" programme that recognises ~180 of the world's top technology visionaries for their proven cross-platform technical expertise, community leadership, and commitment to business results. Steve blogs regularly on DevOps and cloud topics at www.devopsgroup.com/blog/.

Research methods

Developer Velocity

The McKinsey research into the Developer Velocity Index (DVI) takes into account 46 different drivers across 13 capability areas (exhibit). To develop and validate this list of drivers, they conducted interviews with more than 100 chief technology officers, chief information officers, and other senior engineering leaders. They then asked technology executives at 440 large organizations across 12 industries in nine countries to rate their company's performance. DVI scores are calculated as a weighted average of scores across the drivers, with equal weight given to the three broad categories—technology, working practices, and organizational enablement. Their analysis examined the impact of DVI scores on revenue, total shareholder returns, and operating margin. They also looked at four non-financial business-performance indicators: innovation, customer satisfaction, brand perception, and talent management. Finally, they ran statistical correlations of business performance against the various dimensions of Developer Velocity. They then used Johnson's Relative Weights analysis to quantify the relative importance of the correlated drivers of DVI scores. to read more about the DVI research, please click here.

Additional Research

Additional data was collected by surveys and interviews with Sogeti's Cloud & DevOps practice leads. In total these practice leads are responsible for over 250 customer cloud and DevOps implementations and their feedback was used to provide the recommendations and additional customer insight in the report.

Disclaimers

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product.
You may copy and use this document for your internal, reference purposes.