ITIL® and DevOps
Getting started
Kaimar Karu

AXELOS.com
Contents

Executive summary 3
Introduction 3
Part 1: Context and background 5
Understanding the current challenge 5
How did we end up here? 6
Thinking in services 7
Part 2: Getting started with DevOps 10
The DevOps stables: horses and unicorns 10
The three ways: enabling and adopting DevOps practices 12
Part 3: Evolving IT operations 19
Operations as a platform 19
Afterword 21
Appendix A: A subset of DevOps concepts and practices 21
Appendix B: ITIL Guiding Principles 22
Appendix C: Recommended additional reading 23
About the author 24
Reviewers 24
About AXELOS 25
Trade marks and statements 25
1 Executive summary

The DevOps movement presents IT service management (ITSM) professionals with an invaluable opportunity to enhance the support they offer the business by increasing value for the customer through improved services due to better service management practice. At the same time, organizations fully immersed in the world of DevOps have begun to understand the value of the service mind-set, with its immense positive impact on customer value and its benefits to the bottom line of the business.

Roughly speaking, three types of organizations are looking to leverage DevOps practices: the ones born in the world of apps, cloud computing, and automation by default; the ones successfully leveraging new technologies and practices to rebuild their existing IT capabilities; and the ones just starting their journey from the mean-time-between-failure (MTBF) world to the mean-time-to-restore (MTTR) world. There’s a good chance your organization is in this third category. The majority of organizations are! In which case, this guidance paper is here to help you.

While the DevOps philosophy encourages collaboration with the whole organization, including the end-to-end service lifecycle, DevOps practices have been most telling within IT Operations, specifically IT capabilities. This is also where the quickest wins for ITSM are today – and the highest customer value.

This paper explores the context of ITSM in modern-day organizations. It covers the improvement of practices over time and the challenges we are facing today. It then aligns the wider scope of ITSM to specific DevOps practices, and highlights key improvement opportunities, including tips on getting started. Not everything covered in this paper is new but never before has it been possible to perform ITSM work so efficiently.

The last section of the paper provides a new view on the role of IT operations in the modern organization; a role that allows IT operations to fully utilize their expertise and become trusted advisors and true business enablers.

2 Introduction

Increased organization-wide awareness of the benefits of DevOps practices, including board-level visibility, has allowed ITSM professionals to approach their service improvement initiatives with the requisite support from their stakeholders.

DevOps

DevOps (a portmanteau of development and operations) is a philosophy and a movement focusing on organization-wide collaboration to support the delivery of value to the organization and its customers. The drivers for DevOps can be seen in the increasing awareness and acceptance of the complexity of business and technology environments, the rapid developments within IT that make it possible to ‘do the right thing right’, and the overall acknowledgement of the crucial role that people play within this and the value that they bring. These aspects are all aligned with the ITSM mind-set.1

In organizations that have embraced the DevOps philosophy, and have introduced improved ways of working, it is the IT operations team that has so far reaped the most benefit from the emergence of DevOps. This is because DevOps practices strengthen the connection with, and the feedback loops between, the development of a service and its operation once it has been released.
IT operations
The phrase ‘IT operations’, used extensively throughout this paper, signifies a combination of ITSM capabilities predominantly residing in the service design, service transition, and service operation stages of the service lifecycle. It covers processes like availability management, change management, release and deployment management, service asset and configuration management, event management, incident management, and problem management. This is also where DevOps practices intersect with ITSM the most.

DevOps practices leverage various Lean techniques, such as Kanban, to improve the quality of work. They bring together the people who are developing individual components of the service and the people who will run the service, to ensure maximum collaboration and a smooth transition from one to the other.

Lean
The core idea is to maximize customer value while minimizing waste. Simply, lean means creating more value for customers with fewer resources.

A lean organization understands customer value and focuses its key processes to continuously increase it. The ultimate goal is to provide perfect value to the customer through a perfect value creation process that has zero waste.

To accomplish this, lean thinking changes the focus of management from optimizing separate technologies, assets, and vertical departments to optimizing the flow of products and services through entire value streams that flow horizontally across technologies, assets, and departments to customers.²

IT operations, DevOps helps to address the age-old problem of being presented with a service, or a component of a service, that is unsuitable for the job for which it is intended. When software development and IT operations teams work together, the whole lifecycle of a new service, application, or improvement initiative tends to run smoother, cheaper, quicker, with a net result of a better quality, better fit for purpose, better fit for use product, and a more satisfied customer.

Adopting DevOps practices is a transformation initiative for most organizations, as is any large scale improvement. You cannot ‘implement’ DevOps. It is a journey. The journey requires specific enablers to be put in place before DevOps practices can be fully adopted. Many of these enablers are outside the IT operations domain. They cover the whole service lifecycle, the whole value stream involved in delivering value to customers through fit-for-use and fit-for-purpose services.

For some organizations, this seems overwhelming, maybe even impossible to achieve. Organizations find starting this journey to be a significant challenge, and they find it difficult to know where to begin, or how. Organizations that have IT systems older than, say, 10 years cannot simply switch over to new ways of working, to fully leverage cloud computing and/or agile Software Development practices.

Yet, many organizations have already achieved success with adopting DevOps practices. This paper aims to make it easier for organizations to get started.
In Part 1 of this paper, we will explore the journey that organizations have taken when moving from reactive fire-fighting IT processes to a service-based approach of delivering value using IT capabilities, and the challenges this has presented. We will also explore the alignment of software development and ITSM, and introduce key DevOps concepts in the context of ITSM.

In Part 2 we will go into more depth to explore specific DevOps practices, and how to best leverage these to improve an organization's ITSM capabilities. In addition, we will look at ways the service mind-set brings additional value to organizations who are adopting DevOps practices.

Finally, in Part 3, we will finish with a suggestion for how IT operations teams could rethink their role in the wider context of the organization, to become true business enablers for the whole service delivery value chain.
PART 1: Context and background

Understanding the current challenge

In an environment where IT has to focus on keeping the lights on, ready to enter fire-fighter mode at a moment’s notice, many IT professionals are overloaded with a never-ending stream of work. ‘Dark matter’, as it is sometimes referred to, the value and extent of which they find hard to quantify. In trying to address this, organizations often struggle to go beyond the implementation of reactive processes; the adoption of an end-to-end service management mind-set is more than they can hope for. For many organizations, the focus of improvements has been limited to a few capabilities. These are described in ITIL® Service Operation (service desk, incident management, and reactive problem management), ITIL® Service Transition (change management and configuration management), and ITIL® Service Design (service catalogue management and availability management). These are often in isolation from one another and from the rest of the value chain, which can lead to local optimizations that have limited value for the organization as a whole.

The situation is further complicated by the misconception that ITIL – a commonly used framework for ITSM – needs to be ‘fully implemented’, and that its benefits will only be seen if every aspect of every process is in place as described in the ITIL lifecycle suite books, and each role is mapped onto a corresponding, separate, job position. This would be a Herculean task at the best of times, and an unnecessary one at that. Under the influence of these misconceptions, it is no wonder that many organizations have not been able to fully leverage the ITSM benefits.

Initiating improvements in isolation from the wider organization can prompt many challenges. There has been a rapid emergence of new technologies and practices in the past decade. Both software development and IT operations have improved their ways of working. The trouble is, they have improved in ways that now need significant effort to re-align. Partially due to the difference in the way they view and address change, the misalignment of their practices has grown considerably over time.

The pain caused by this misalignment has given rise to the DevOps movement. DevOps is underpinned by a philosophy that transforms the way IT professionals think about the stability and resilience of the systems they manage, as well as about their job roles in the end-to-end view of value flow. The ‘development vs. operations’ mind-set is disappearing, albeit slowly, as both sides have come to understand that they are on the same team and are working towards the same goals.

It is important to remember that some often quoted metrics with regards DevOps are not necessarily where the focus should be. The number of software deployments per day can provide interesting
insights, but the capability to deliver value faster is what matters. From a similar viewpoint, using cloud services is not about the cloud per se, but about leveraging the technology available today to rebuild existing IT infrastructure into something much more resilient, improving specific work practices in the process, and supporting the whole organization by doing so.

In the traditional IT operations world, some common DevOps practices might seem incompatible with existing strategic objectives and ways of working, especially when it comes to planning, service deployment, and risk management. While not entirely novel for high-performing organizations, some of these practices would have been unfeasible for most organizations in the past. The opportunities presented by today’s technology have made it achievable for most.

For example, the provisioning and delivery of computing power for cloud-based solutions has now been commodified and often automated. Hardware is treated less like a collection of artisanal artefacts and more like electricity – ‘pets vs. cattle’, as it is often referred to. Agile software development practices, the capabilities of modern IT infrastructure management, and automation have made it possible for IT professionals to truly become business enablers. IT is no longer there just to keep the lights on. Instead, it provides great value and significant advantages for the organization and their customers over their competitors. Furthermore, as this value has become apparent throughout the organization, non-IT departments are increasingly open to including IT capabilities as part of their strategic assets, rather than as a support function. In other words, true ITSM capabilities are now attainable for most organizations, if they are interested, and motivated to do so.

The challenge for most organizations is to understand the practicalities of how to make it work. It is important to understand how technology, including cloud computing, has changed over time. The cloud of today has progressed, in its current form cloud computing is an enabler and a prerequisite for most DevOps practices. The DevOps movement has helped to combine the ‘what’, ‘why’, and ‘how’ of leveraging IT for organization-wide value delivery, focusing strongly on the ‘how’, and therefore providing an invaluable resource for ITSM professionals.

How did we end up here?

For software development, change has always been the norm. For IT operations, change has always meant additional stress on people and systems.

Some time ago, when the IT departments of most organizations were struggling with fragile in-house infrastructure, the primary focus of IT operations teams was to keep the systems running with little time left for other tasks. The maintenance of the production environment – the most visible part of the IT infrastructure – required a lot of (manual) work.

On the software development side, both the functional and non-functional requirements kept changing throughout the design and development phases, which led to mid-flight adjustments to code and (sometimes significant) changes to the infrastructure. This in turn led to various ad hoc changes – patches - in the production environment for both software and hardware, which were deployed by IT operations. This made clean redeployment virtually impossible, and led to further ad hoc changes when something broke, increasing the fragility even further.

High cost of hardware and long procurement times prohibited the utilization of identical development, test, and production environments. As a result, the code that was released had often been tested with a smaller data-set, using considerably lower utilization levels, and completely different infrastructure. If it had been tested at all, that is. The demand for swifter software delivery was already there, and often the only way to speed things up was to decrease testing time. We have all experienced the results of these shortcuts.

For developers, it was almost impossible to know how their code would perform in the production environment, so guesses had to be made. Long deployment cycles removed developers from the frontline, and tracking the reasons for a failed release six months after the code was delivered was nearly impossible. Similarly, it was difficult to predict the software’s demand on hardware capacity for (at least) a tolerable user experience, leading either to serious capacity issues or significant overspend on hardware.

Organizations, suffering from these underperforming IT systems, were not happy. Work was undertaken to identify common successful practices for developing and managing IT systems, and proposing new ways of working where the organization’s existing practices didn’t lend themselves to significant improvements.
On the IT operations side, this work focused mainly on two aspects: infrastructure and customer service. The former covered both the hardware and system management software, and the latter dealt with the two-way interaction with the customer. On one hand, it was clear that IT had to better understand the organization’s requirements for new IT systems, and on the other, user experience of IT systems had to be improved.

In the 1990s, software development professionals were actively searching for viable alternatives to the waterfall approach, new ways of working to support incremental development methods that could allow applications to deliver value more promptly and more frequently. While some of these practices had first appeared in the 1970s, they had not yet entered the mainstream. The search led to the introduction of extreme programming, and the rise of various other agile methodologies.

At the same time, IT operations professionals were looking to improve the customer’s overall experience with IT systems. The objective was to allow business units to focus on their core functions without the need to familiarize themselves with the specifics of various technologies, and without being impacted by the performance issues of IT systems.

Rather than exposing the ‘back of house’ directly to the end users, IT systems (including all hardware and software) were packaged into services, and IT’s ‘front of house’ (in the form of a Helpdesk, or a Service Desk) took on the daily interaction with the user. This field of expertise, dealing with IT services, became known as ITSM, and was codified in the second version of IT Infrastructure Library – or ITIL for short.

In the coming years, it became abundantly clear that focusing on IT operations was not enough to design, deliver, and support services. The experience of both public and private sector organizations around the world had shown that IT capabilities had to be aligned with the organization’s strategy earlier than the execution or maintenance stages. An updated understanding of what IT means for a modern organization was required.

When ITIL underwent a major revision in 2007, and was refreshed in 2011, dropping the long form of IT Infrastructure Library, it mapped proven IT practices onto a service lifecycle approach. This covered the strategy, design, transition, and operation of services, enabled by continual service improvement – or kaizen as it is known in the world of Lean. The updated approach promoted an end-to-end service view, including the development of the software components of the service, and focused on leveraging IT capabilities to enhance the overall business value, rather than IT functioning in its own little world, ‘doing IT’.

The growing complexity of technology and practices in both IT operations and software development kept most professionals focused on their domain, and rarely was there a chance to look up and focus on the whole value stream, or seriously analyze the value the customer received from IT.

Over the past decade, the advancements in IT have made it significantly easier to align the efforts of software development and IT operations. Cloud computing, drastically changing the cost and delivery times for additional hardware capacity and software updates, has simplified many previously laborious activities. Both sides of the historical development/operations divide have realized that the only way forward is through close collaboration.

To make full use of these new capabilities, it is important to review existing ITSM processes and procedures in order to ensure IT operations don’t become a bottleneck in delivering additional value to the organization. For IT operations professionals, this opportunity for revising how work is done is a relief – much of the once-required, perhaps even previously preferred bureaucracy (or ‘scar tissue’, as Adrian Cockcroft refers to it) has become unnecessary. Effort can be re-focused on what matters today and what will provide a competitive advantage tomorrow. Large sections of the ITIL guidance that were once difficult to adopt, e.g. automating standard changes and enabling continual improvement, are now within the capabilities of most organizations. This is where various DevOps practices can be successfully leveraged.

5 Thinking in services

In the early days of ITSM, the main discourse was on ‘IT and business alignment’, with the focus on ensuring IT wouldn’t be a siloed function in the underbelly of the organization. A decade later, when IT had become an integral part of many organizations’ value streams, the discourse shifted to ‘IT and business integration’, to better leverage technology for business purposes. In both cases, there was a clear acknowledgement that IT had historically stood separate from the ‘core’ of the business. For IT departments, working towards alignment with the rest of the business while aiming to become a recognized (and equal) part of the organization was an invaluable interim step.

The concept of IT services was used to describe the mechanism of delivering value to the customer – both internal (other departments) and external (end customers). This was a significant evolutionary step away from the technology-focused mind-set that assumed IT was just there to ‘do IT’. Information technology is only one of the elements required for a successful service, and often not the only type of technology, either.
Figure 5.1 A common (pre-DevOps) view of software development and ITSM. The dashed elements represent capabilities that are often invisible, or assumed not to be there at all. The solid-colour elements are where most of the visible work happens. From these, the orange-lined elements represent capabilities that are sometimes seen to be the main source of the pain when it comes to collaboration and value delivery.

**Definition: Service**

A service is a means of delivering value to customers by facilitating outcomes that customers want to achieve without the ownership of specific costs and risks.

For some time, the capabilities required to enable end-to-end service management have been only partially addressed, with a heavy focus on reactive processes, and with often only a perfunctory acknowledgement of strategy and collaboration. This is not because IT professionals didn’t want to do more or to do it better but, within the constraints of limited resources, anything beyond what was needed for survival was not regarded as priority one.

Times have changed. Today, much more is possible. The DevOps movement highlights some of the key areas where significant improvements can be introduced. In many cases, the new practices help to overcome the constraints that have prevented ITSM professionals from utilizing their specialist skills to realize the true potential of IT services. In that sense, DevOps is an opportunity for a significant continual service improvement (CSI) initiative for IT service providers.

While collaboration is highly encouraged in the DevOps philosophy, people working in software development and IT operations are not necessarily aware of each other’s goals, practices, and challenges. Figure 5.1 describes the often occurring misalignment between software development and ITSM. The building blocks in this model are not meant to signify individual teams – this is a view of capabilities and alignment, not an organizational model. In here, the ITSM domain is limited to a subset of IT operations, mostly reactive, focusing on aspects of service transition and service operation. The main points of connection between software development and ITSM are those many people associate with pain: change management, including the sometimes dreaded Change Advisory Board (CAB) meetings; and release and deployment management, which, when introduced in practice, often brings long delays and manual deployment tasks.

The challenging perception is that software development focuses purely on new functionality, with little regard for testing, operational aspects, or long-term supportability of new releases. Code is often ‘thrown over the wall’ from the Build environment to release management, to be installed by someone from IT operations over the weekend. The stereotype of hot-headed developers and slow-paced operations specialists is common in this world.

With the advent of agile development, the value stream’s constraint was moving from software development resources to IT operations. Developers and IT operations specialists could no longer afford to continue to ignore each other, or to leave the conversation to heated discussions during marathon CAB meetings. Both
sides wanted the other side to raise their game and take responsibility, and it turned out both sides were quite willing to look at ways their work practices could be improved. Maybe there weren’t any ‘sides’ after all. The problem was less about improving the two functions independently, and more about the interactions between the people, processes, and practices, improving the system as a whole.

The end-to-end approach started to receive more focus, and various continuous integration and continuous deployment practices were introduced, enabling more natural ways of collaboration, higher levels of automation (to a point), experimentation, and quicker delivery of new functionality.

This is only part of the story. The challenge for most organizations is not just how best to introduce DevOps practices, but how to do this while embracing an end-to-end service view. The idea that we can continue talking about ‘IT and the business’ is dangerous—it continues the separation that has historically defined the relationship between these two. In this environment, DevOps practices will be of limited value for the organization, most likely leading to local optimization. Various constructs, like ‘BizDevOps’ and ‘DevSecOps’, have started to pop up. These are good at making the point that there is more to the end-to-end service than the work that takes place in (traditional) software development and IT operations teams, but there is a limit to the number of abbreviations that we can daisy-chain. The deeper we dig into what is meant by these constructs, the closer we get to what resembles the ethos of ITSM. Yet, the language used is different to what ITSM professionals are used to. To understand how to align these worlds, we need to do a little rethinking before focusing on DevOps.

The perception of service management needs to change. The business alignment and integration initiatives were significant waystations for IT, but a fundamental change is required. IT must stop merely taking orders; it is not enough to focus solely on reactive capabilities, no matter how green the SLA reports.

Trained and experienced IT professionals are usually well equipped to find the ways technology can best support the business’ objectives—not just to satisfy the requirements of today, but to be a part of creating, executing, and supporting the strategy for tomorrow. IT capabilities have become an integral component in achieving a competitive edge, and IT has reached the maturity to become a trusted advisor.

The value to users and customers continues to be delivered through services. The post-deployment domain in IT operations involves at least as much intense activity as the pre-deployment domain of software development, and both sit within a non-technical eco-system that exists to serve. This reality is sometimes forgotten.

DevOps has brought significant improvements to the maturity of practices within service transition and service operation, and the shift in the mind-set to collaboration-by-default has created an opportunity to extend the improvement to the other stages of service lifecycle, where DevOps practices are currently less mature.

---

**Definition: Continuous Integration (CI), Delivery (CD), and Deployment**

Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

Continuous delivery is a DevOps software development practice where code changes are automatically built, tested and prepared for a release to production. It expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When continuous delivery is implemented properly, developers will always have a deployment-ready build artefact that has passed through a standardized test process.

With continuous deployment, revisions are deployed to a production environment automatically without explicit approval from a developer, making the entire software release process automated.
Figure 5.2 shows what might happen if DevOps practices are introduced in isolation from ITSM, while highlighting some common anti-patterns to successful adoption of DevOps practices. Even though some aspects of service transition and service operation are acknowledged, they are still seen as external to the CI/CD pipeline, so ITSM processes like change management and configuration management are not accommodated, resulting in duplication.

The Software Development lifecycle has benefitted from DevOps practices and is more mature, but on many occasions where agile development practices are used, the expectation is that product strategy will be taken care of elsewhere in the organization. This reduces the development teams to the status of order-takers, and prevents them from maintaining an integral role in the decision-making process. The practices of continuous integration and continuous deployment support the iterative development approach, but the feedback loops between software development and IT operations are still incomplete. The same applies to the feedback loops between software development and the end user, as functions like the Service Desk sit outside the DevOps domain in this model.

When looking at this model, a question arises: when exactly is the value delivered? Is it after each deployment, when the new functionality is in place, or is something more needed for the customer to receive the expected value? While the expectations and available financial levers of both users (the ones who use the service on a daily basis) and customers (the ones who pays for the service) are different, both are interested in more than just functionality. They are interested in an end-to-end experience, and in achieving their objectives. We need to keep this in mind as we start exploring the world of DevOps.

Functionality and service

Uber has made it possible, some time ago as part of their ongoing development, to log separate credit cards for personal or work-related trips, which simplifies expense reporting, and makes it easier to distinguish between personal and business expenses.

Another part of their functionality, at the core of the platform, is that customers can easily notify the Uber team through the app if they have a problem with the ride. An Uber employee will review the feedback, take action if required, and get back to the customer.

Adding such functionality is guided by Uber’s product strategy, and then executed in the app by their software development teams. However, the holistic customer-focused solution requires more than just the functionality in the app.

As paying customers, people using Uber have expectations for car availability, driver skills, overall safety, and customer support. They expect a particular level of service. While these expectations are often fulfilled with the help of technology, software development is only a part of the end-to-end value chain.
Part 2: Getting started with DevOps

The DevOps stables: horses and unicorns

Thanks to people like Patrick Debois, Andrew Clay Shafer, and many others who refused to accept the status quo of the last decade’s disjointed approach to IT, DevOps was born. The word itself, formed from ‘dev’ for ‘development’ and ‘ops’ for ‘operations’, signifies a philosophy and a movement focusing on organization-wide collaboration to support the delivery of value to the organization and its customers.

The key drivers for DevOps were the increasing awareness of the complexity of business and technology environments, and the rapid developments within IT that make it possible to ‘do the right thing right’, alongside an acknowledgement of the crucial role that people play within this and an appreciation of the value that they bring.

The DevOps philosophy focuses on five highly integrated areas, covering values, behaviours, patterns, practices and techniques in Culture, Automation, Lean, Measurement, and Sharing (CALMS).

CALMS

Culture: supporting collaboration by addressing values, beliefs, attitudes, policies, and informal structures in the organization that influence decision-making and ways of working

Automation: replacing repetitive tasks with automation to ensure consistency and free individuals’ time up for higher value-adding activities; for example: automated tests, quality gates and deployments

Lean: understanding the value chain and optimizing for flow while avoiding local optimization, supporting continual improvement (Kaizen) and reducing waste (Muda)

Measurement: collecting data, ensuring visibility, and enabling learning while making sure the right level of data is captured and analyzed

Sharing: feedback mechanisms within and between teams, especially between software development and IT operations, including both left-to-right and right-to-left feedback loops
Significant effort has gone into keeping the definition of DevOps as open as possible, by refraining from defining or constraining it too soon in its development. This approach, closely followed for nearly ten years, has allowed the movement to flourish, but it has also brought some challenges.

For some organizations, DevOps is the only lens through which the world is, or has ever been, seen. In DevOps-related discussions, these organizations are often called 'unicorns', because the practices they commonly use can seem magical in comparison to the practices of others. These organizations have, in most cases, formed within the past ten years, and they have six significant, differentiating characteristics:

- An IT infrastructure built in the cloud by default
- Agile Software Development practices used by default
- Business transactions take place online
- Success measured primarily not on revenue or profit, but on the acquisition and retention of users
- The products and services their users consume are not necessarily the same products and services their paying customers are interested in
- The organizational structure is built around products and is supported by cross-functional teams without rigid walls between departments.

These organizations tackle legacy systems in their own distinctive way, and by 'legacy' here we mean software that has been in use for just a few years, not for decades. Change is the norm; changing everything from the cloud service providers to the programming languages they use is written into the organization's DNA.

A second group of organizations, still a relatively small one, is sometimes described as ‘horses with horns’ – traditional enterprises that have successfully re-architected their entire IT systems to take advantage of today’s technologies. They are either operating in a way very similar to the unicorns, or, in some cases, they lead the way in the introduction of new DevOps practices.

The majority of organizations are seen as just ‘horses’ – they may be thoroughbred, but they are without the magic of the unicorns. In these organizations, the investment in IT infrastructure tends to go back more than ten years, the practices in place have been optimized for managing expensive and always-failing infrastructure that took months to procure, and the experience with new software releases has often been intensely painful, leading to expensive change programmes instead of a continuous flow of small, incremental changes.

For these organizations, change is something that is considered unwanted; slowing down the rate of change was believed to equate to a more stable IT infrastructure, and therefore more stable services. And, by extension, less dissatisfied users.

These organizations have often optimized their IT capabilities for what was possible 10, 15, or 25 years ago. In today’s world, the pressure on them to change their practices is immense but, rather than rebuilding everything from scratch, the improvements have to be addressed step-by-step so that the system does not break.

An important element of this phased approach is the focus on people. The widespread acceptance of the DevOps philosophy has changed the way the industry thinks, and this has allowed us to pay attention to the people-side of IT.

We, as an industry, are now ready to discuss organizational culture, organizational change management, and people’s role in organizational success. We don’t expect documented processes, written procedures, and advanced toolsets to deliver value on their own. We know that, in the centre of value delivery, there are people who should be trusted and respected.

The three ways: enabling and adopting DevOps practices
The adoption of DevOps practices is a journey. For some organizations, this is the same journey they would expect to take as the company grows and improved working practices become available to them. For others, it’s a tough uphill trek that requires special preparation.

On this journey, the challenges most organizations encounter are similar. The following provides guidance on how to start tackling some of these challenges. The structure echoes the structure of the Three Ways, as described in The DevOps Handbook and The Phoenix Project, which are important books for those exploring the concepts of DevOps:

- The first way: systems thinking
- The second way: amplify feedback loops
- The third way: culture of continual experimentation and learning.

7.1 THE FIRST WAY: SYSTEMS THINKING

Systems thinking deals with complex adaptive systems, and focuses on interactions rather than on the individual components of those systems.

**Definition: Complex adaptive system**

A system where complex non-predictable behaviours emerge as the result of the interaction between the components within the system, which themselves can be complex adaptive systems, and from the interaction with other complex adaptive systems.

Systems thinking brings the focus back on the performance of the system as a whole, rather than on the performance of its individual components in isolation. This doesn’t mean that the efficiency of individual components is not measured or improved where necessary, but no one component is optimized without considering its place in the whole system. The value stream doesn’t cover merely the path from development-to-operations, but instead looks at the entire service, from business strategy to customer experience.

Those familiar with ITIL might recognize this as the service lifecycle, which covers the following:

- **Strategy:** who are the customers and users, and what services do they require to achieve their objectives?
- **Design:** how should the services look and feel, and what capabilities are needed to provide them?
- **Transition:** how to develop or procure, test, and release these services?
- **Operation:** how to support services in the production environment, and how to provide a great customer and user experience?

All of this is supported by continual service improvement, which enables the various feedback loops between different stages of the lifecycle, between processes, and between the parties involved in the value chain.

Appendix B: ITIL Guiding Principles, contains more information on how systems thinking influences the way ITIL should be adopted and adapted.

7.1.1 Understanding value

The driver for all improvements is customer value. It is the customer who defines what is of value to them, not the service provider.

This is not to say that the service provider should be a mere order-taker, waiting for the customer to hand down decrees and requirements which the service provider will then obediently fulfil. The value, while supported by the service provider’s specialist capabilities, is created in collaboration with the customer. An appropriate analogy is the model of the trusted advisor – with the added benefit that the advisor is also able to execute what was proposed, discussed, and agreed.
It is important to ensure that the strategy for services is developed to support the business’s objectives, and that the services themselves are designed in close collaboration with the customer. While using technology in insightful ways can provide a significant competitive advantage for the whole organization and its customers, IT service providers cannot create their strategy in isolation from the rest of the organization, especially considering that most business functions are heavily dependent on IT. ITSM will have only limited success if it is seen and behaves as an ‘IT thing’.

The optimization effort should primarily be targeted at the customer experience of the service, rather than toward improving the ease with which the service provider manages the service. In addition, ITSM professionals need to make sure the language they use with their customers is not overly technical. A common language is invaluable in building a trusting relationship, yet highly specialized language is unlikely to be understood by all parties. In many cases, especially with internal service providers, adopting the customer’s language and providing guidance where a common glossary doesn’t exist will be more successful than enforcing ‘IT speak’ on the customer.

7.1.2 Removing silos

Close collaboration with the customer is vital, involving each team who has responsibility for delivering components of the service. Sometimes, it makes sense to bring those teams together to work as one team, though other times co-location works fine, too. While specialization is usually necessary, it is important to remember that ITIL considers all ITSM roles as just that – roles, not positions, and certainly not silos. Each process has been put in place to support the organization’s capabilities, and the people responsible for those processes need to look beyond their own teams and specific remit to maximize value.

It is unrealistic to build a successful service management capability if people involved in different parts of the value chain don’t talk to each other, or when they base their activities on conflicting objectives, priorities, and incentives. There is usually no need to go so far as to set up a separate committee or board to facilitate discussions between teams. When the ‘What’s In It For Me’ (WIIFM) question is adequately addressed, communication follows naturally; much of the ongoing communication will be informal, rather than in meetings and documents.

In some organizations, there may be a common team who look after the majority of the ITSM capabilities and processes. In other organizations, there may be individual teams assigned to individual processes. Regardless, it is important to ensure all teams communicate with each other. If, for example, the incident management team steers clear of the problem management team, or if the change management team have hidden themselves away from everyone else, then the likelihood of delivering a high value service is low. Close collaboration between teams must be encouraged, (though to stress, this is not to discourage the use of specialized teams, if they are appropriate). If working together closely on a daily basis is not feasible, then the teams should at least be in a position to understand what the other teams are doing, both upstream and downstream, and how the work and information flows between them.

7.1.3 Unifying the toolchain

One reason that silos emerge is the different toolsets that different teams utilize. While developers might use Jira or LeanKit to manage their workflows, for example, IT operations professionals may use their own tools like Remedy or ServiceNow, which could mean duplicate records are needed for any given task so that the work can be tracked using these different toolsets.

On a more technical level, one of the common challenges for ITSM is the configuration management system (CMS). It is rarely up to date and a CMDB is extremely difficult to manage when changes are logged long after they happen. If software developers use practices like continuous integration and continuous delivery, then it is possible they have built a system that can be integrated into the CMS, or
perhaps can even become the CMS for IT operations.

It is a good idea to review the tools each team uses to ensure there is no unnecessary duplication. It might not be possible to utilize the same tool for both development and operations, but, at a minimum, multiple ‘sources of truth’ should be avoided, especially when they are potentially conflicting. In such situations, one tool can be designed to act as a central source of information, which the other tools can reference, rather than create duplicate data sets. It is also useful to assess which manual activities can be automated using the current toolset, or with the use of additional tools. One of the areas where automation can provide instant value is change management, leveraging the speedy implementation of standard changes. (A quick heads-up – buying in an expensive automation tool might not be the best first step to take – there are usually quicker wins).

7.1.4 Managing risk, assurance, and compliance

Another key area that benefits from a holistic view of the service lifecycle is regulatory compliance. Many controls – or barriers, depending on the viewpoint – have been put in place in IT Operations to ensure that the compliance requirements for, for example, SOX and PCI DSS, are accounted for. Some controls that have been put in place may be strategic constraints, and highly valuable, while others were introduced as ‘patches’, but weren’t valuable in the long run.

Some of these controls have, in turn, added more controls to other stages of the service lifecycle, and have a significant impact on software development. As many controls have often been developed in isolation and then enforced upon the relevant teams, rather than co-developed, they can lead to a lot of waste in the system due to the lack of visibility of objectives, and the additional workloads the controls bring.

An automation-heavy, platform-based approach that enables - no, enforces - compliance requirements, such as separation of duties, considerably improves the service provider’s capabilities to deliver world class services while following all relevant regulations. Continuous delivery is a key practice here.

An effective CI/CD pipeline does not mean that developers need to log in to production environments to deploy new code. Quite the opposite. While the release package flows through the pipeline, a series of pre-defined automated quality gates ensure defective (or malicious) code doesn’t reach the production environment.

Another benefit of automating processes is the improved traceability it brings, compared to the manual release and deployment process. Everything that is done to or within the system is automatically logged the moment the activity is carried out. No more searching the logs or relying on memory for ‘who did what and when’; auditable log files are created by default.

This added capability is new to most IT auditors. It is advisable to work closely with the auditors to find out what information they require, and in what format. A successful collaboration here will significantly decrease the workload for all parties involved.

7.2 THE SECOND WAY: AMPLIFYING FEEDBACK LOOPS

Although most of the work flows from ‘left to right’ (i.e. from business strategy to software development to IT operations to customers), all components of the value stream must provide feedback to the components left of them. Feedback loops are crucial in DevOps.

7.2.1 Left-to-right and right-to-left feedback

Continuous feedback like this can be enabled through CSI and is fundamental for successful service delivery. While, for example, strategy should be agreed before a new service is commissioned, these agreements should be treated as enabling constraints, open to review as required. When downstream service components report challenges or inefficiencies, this should be taken into account when strategy is reviewed. The same applies to service design and transition models and practices, which can benefit from feedback
from both IT operations and customers.

Feedback loops can support both staying on course and choosing a new course. Therefore, revisiting aspects of the strategy should not, in most cases, be necessarily seen as a major change initiative. It can be part of normal work. As for other aspects of services, it is highly likely the blueprint and components of a service, including all the code, processes, procedures, and tools required, will change as the development commences, and again after the service has been deployed. Passing through design (and then through transition back to operation) to address a deficiency discovered in service operation is perfectly fine, and is to be expected. This is a benefit of monitoring – both the use of monitoring tools and the direct observation of customers, partners, and other teams working in the value stream.

Developers should be involved in the design stage for all services that have custom-developed software components – which in today’s world could mean most, if not all, IT services. An iterative approach to service design and transition can identify issues with the functionality of new services, as well as with the state of user readiness and system operability. The closer the collaboration, the less likely it is for issues from design and development to reach the production environment.

If the organization develops software in-house, the ITSM professionals need to talk to developers before, during, and after new services are added or existing services are changed. In many organizations, the software is brought in - either as COTS (commercial, off-the-shelf), or with outsourced development. In these cases, the ITSM professionals should be in a constant dialogue with e.g. the PMO (project management office), or with the team that manages the outsourcing partners.

### 7.2.2 Improving capabilities

For many organizations, introducing CSI as a key capability in ITSM could be a reasonable first step towards improved feedback loops. ITIL Practitioner goes into more detail on how to approach CSI at different levels in the organization. Some organizations who have introduced CSI in the past might be in a situation where the benefits of CSI have not yet been fully realized. In most cases, this has to do with the speed of feedback, especially the right-to-left type.

‘Traditional’ CSI, if introduced through formal meetings and kept at strategic level, can be too slow for everyday learning. Feedback mechanisms should be enabled on various levels, not just strategic. There are several key processes and functions in ITSM that can be improved to provide instant feedback, and help with improving the quality of services, often instantly.

When it comes to user feedback, the Service Desk can act as a central point for collecting and filtering feedback about potential improvements. The Service Desk is in a unique position to access feedback across all services and service components, which can help to create a holistic approach to improvement. Frequently asked questions can be used to improve documentation, or, if applicable, the design of the service, to remove the need for further clarifications. Having a Service Desk does not negate the need for the product or service team to have direct contact with the user community in all other stages of the service lifecycle, though. It is only one of the channels – albeit the one most utilized when a service failure occurs. Specialized, customer-focused skills within the Service Desk team can be invaluable when other teams are busy fixing the outage.

Some processes are designed to act as feedback loop enablers by default:

- Event management, when automated, can provide instant feedback about the success of a newly released service, and may possibly act as a trigger for automated incident resolution, depending on the level of instrumentation in the system.
- Problem management can analyze recurring incidents and help identify the root causes, and various data collection and analysis tools can help automate parts of this.
- Pattern analysis in demand management can inform about trends in capacity requirements and help keep down the costs of e.g. the cloud infrastructure.

### 7.2.3 Course correction
Improved feedback loops can also be crucial for the survival of the organization. In the start-up world, where new markets are constantly discovered (or created), there is more flexibility in how to react to feedback. Some organizations have the ability to pivot – change their core business model when the current one doesn’t fulfil their needs, or doesn’t match with the capabilities they have to serve their customers. With start-ups, this might happen a few times during their life. In established organizations, reacting to feedback takes longer, and a complete pivot is an unlikely solution for most. Having access to continuous feedback and being able to detect weak signals can be invaluable when making strategic decisions.

7.3 THE THIRD WAY: CULTURE OF CONTINUAL EXPERIMENTATION AND LEARNING

Each stage in the service lifecycle should interact with all other stages at all times, both directly and through continual service improvement (CSI), enabling experimentation, learning, and re-alignment.

The word ‘culture’ in the context of IT can be rather esoteric and it is often left undefined. Is it about individuals or the whole organization? What constitutes culture – does it only cover organizational value, or can we include ways of working? What about tools? The choice of a cloud platform, data store model, programming language, etc. are components of the ‘how’ of a service provider’s answer to their customers’ needs: is that also part of culture? The short answer is that organizational culture is no one specific thing, and it’s as much about individual elements as it is about interactions between these elements.

7.3.1 Experimenting

In the context of complex adaptive systems, it is rarely possible to predict exactly how situations will work out. Though people working in IT are used to pockets of causality, it is risky to apply proven small-scale models to large-scale systems. Usually, we don’t know enough about the myriad forces and interactions to accurately predict the performance of large-scale systems. We need to learn more about how the system works, and approach the task with small steps, which can take the form of experiments.

In some ways, experimentation in IT is not uncommon – but it often resembles ‘hoping for the best’ rather than structured, controlled experimentation. It is essential to be aware of what is being tested – formulate a hypothesis, design an experiment, and learn from its results. Experiments help us to understand systems and, even if a hypothesis is falsified, there will always be something that can be learned from it. Learning from failure is an important element of experimentation. (Parallel) safe-to-fail experiments are advisable, wherever possible. There is rigour and discipline involved in experimentation in IT, as there is in science. The scientific method can be a suitable approach.

Definition: Scientific method

Principles and procedures for the systematic pursuit of knowledge involving the recognition and formulation of a problem, the collection of data through observation and experiment, and the formulation and testing of hypotheses (Merriam-Webster).

The following steps, taken from ITIL Practitioner Guidance (section 3.2.4.5), give a quick overview of what the scientific method might look like when applied to IT:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Analyse the situation</td>
</tr>
<tr>
<td>2</td>
<td>Formulate a question</td>
</tr>
<tr>
<td>3</td>
<td>Research the field</td>
</tr>
<tr>
<td>4</td>
<td>Formulate a hypothesis</td>
</tr>
<tr>
<td>5</td>
<td>Design the experiment</td>
</tr>
</tbody>
</table>
7.3.2 Increasing resilience

A continual experimentation and learning cycle can help increase the resilience of systems. While some organizations have near-perfected the art (e.g. Netflix Simian Army), for most, it’s about increasing resilience through continual improvement. Safe-to-fail experiments are useful here; it is better to break the system in a controlled, known environment, rather than wait for the customer to discover a weakness in the system in the process of performing a business-critical operation.

Pursuing increased resilience is easier when the infrastructure is in the cloud – whether public or private. With the support of cloud computing, it’s easier to simulate outages – from the loss of one technical service to the loss of a server, a datacentre, or the whole region.

Even with data centres on the premises or a small server room with limited virtualization capabilities, new architectural models e.g. microservices, can begin to build resilience.

7.3.3 Changing culture

To enable experimentation and learning, the pursuit to improve must become a part of everyday practices. Without it, the constant firefighting (be it outages or conflicting project priorities) will push improvement work down the priority levels, which might mean the accruing technical debt eventually manifests itself as an incident, or worse, a full outage. Management has a key responsibility here to enable and encourage experimentation, and to support learning. This includes addressing the fear factor that prevails when things don’t work out as expected, and when assigning blame is the default solution.
Definition: Technical debt
The amalgamation of technical shortcuts and workarounds, which are frequently poorly designed due to lack of time, which were taken to keep an IT system up and running but which impact on the complexity and maintainability of the system, and the effort required for future developments. In Software Development, often solved by code refactoring (i.e. code redesign or restructuring).

Perhaps counterintuitively, discussions of culture are where discussions about tooling are required. An organization is a complex adaptive system and, while it is not wise to attempt to 'change people', which rarely delivers the required results, it is possible to influence the culture by providing opportunities for better ways of working.

Generally, people want to do the best they can. They make decisions based on the available information and, when doing the right thing is too difficult, they look for easier ways to achieve the required outcomes. They reach for workarounds, which might conflict with the goals of the organization. It is not that they want to cause harm, but the system around them may have been built to inadvertently encourage harmful behaviour, or perhaps positive behaviours are discouraged. The way the system works has an immense impact on the way people behave – in a team, in a department, and across the organization. Fixing the system is the only viable and sustainable solution.

Improved monitoring, testing, and deployment tools can make a big difference. Scripting skills have always been invaluable for good IT operations specialists. Yet it is one thing to create your own ad hoc solutions; it is quite another to use a powerful platform to get the job done.

Cloud computing is an example of removing a layer of complexity and replacing it with more simplified services. While all the components (i.e. hardware, software, facilities, electricity, etc.) required for delivering services to the end customer are still there, the management of those components is distributed and can predominantly be outsourced.

Therefore, the focus of the main service provider can be redirected to making the whole service experience better, rather than worrying about the technical aspects of individual service components. With the continuing popularity of the cloud model, service providers themselves are becoming more and more reliant on other service providers. The approach is not new, but the scale of it definitely is, as is the business potential of these capabilities. This has a significant impact on, for example, supplier management practices.

Regardless of the chosen technical architecture – be it Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Backend as a Service (BaaS) – the remaining constant is the customer, who simply wants to get something done. Understanding what that something is, and how to best deliver that capability, is the key.

A change in the organization's culture is not a project. It is a journey. Further, unless it is aligned with the organization's vision and people's incentives, the whole system will fight back. Successful change requires full management support. Yet we must understand that if doing things differently was easy, the management team would have done it a long time ago. Organizations should not be afraid of seeking help from outside professionals as facilitators for change.

Management can be an enabler for change, but quietly waiting for management to change everything for the better is futile. Instead, improvements are more likely to succeed if they are also addressed on an individual and team level, as part of everyday work. It isn't something that can be scheduled for 3 PM to 4:30 PM every second Wednesday. Only by making improvement a standard component of work practices is enough power given to management to steer the organization in the right direction.
The CSI approach is a step-by-step model for addressing an improvement of any size, ensuring it is aligned with the organization’s vision and that it ultimately contributes to the delivery of value to the organization’s customers, as can be seen in Figure 7.1: Putting new structures and new ways of working in place is difficult, and learning new skills takes time. With enough people involved, this might feel like a large initiative with many prerequisites and, as the common state for most people is ‘very busy’, it is often abandoned before it has a chance to get going.

To make it easier for ITSM professionals to begin their improvement initiatives, and to have a ‘true north’ visible at all times, the recent ITIL Practitioner publication describes nine guiding principles (See Appendix B: ITIL Guiding Principles) that can be used for running improvements on a daily basis. Following these principles when dealing with cultural change helps to improve the planning and execution of organizational change management (OCM) initiatives. It is important to note that one does not simply change culture – it’s the way people work and behave that changes over time and, with that, the overall culture of the organization changes, too.

Individual improvements should be kept relatively small, where possible. While knowledge, skills, and experience can be used to find better ways for performing tasks, there is often no single clear answer to the challenge at hand. In these situations, experimentation is the best way to eliminate guesses, and as discussed, the power of technology provides substantial support for this.

For example, if you had to build an actual data centre to test a theory of the value of using a regional data centre, the experiment would be very expensive. Redirecting a subset of the traffic to a cloud service provider’s alternative data centre is a more realistic option to test the hypothesis.

Experiments should be planned in consultation with all impacted teams, as the potential improvements are likely to involve stakeholders throughout the value chain. When working on improvements in isolation, there is a likelihood of local optimization which, in many cases, can be detrimental to the whole system although they appear to be delivering results at a local level.

In any sizeable organization, it is possible that many parallel initiatives in many different places are attempting to address the same issue at any one time. The number of improvement initiatives that are chartered to address the same challenge could be in dozens, if not hundreds across all departments. Many organizations waste enormous amounts of time and money on parallel isolated initiatives that are often driven by conflicting incentives. Finding out which improvement initiatives are running or have been planned, and aligning the ones with similar objectives – and cancelling clear duplicates – can provide savings in millions.
Part 3: Evolving IT Operations

Operations as a Platform

Considering all the focus technology has received in this document, and all the wonderful things now possible through cloud computing and automation, a reasonable question arises – what will become of IT operations? Is there even a place for IT operations in the world of DevOps?

Many of the traditional IT operations tasks, the ones most visible and relevant for software development – procuring new servers, installing operating systems and supporting software, deploying new versions of developed software – can, in theory, all be automated. In some organizations, they are already automated their way to being. Perhaps the over-the-wall dilemma will soon be resolved because there is no one left on the other side of the wall?

A WORLD WITHOUT SERVERS

For the past few years, there have been discussions about an operating model referred to as ‘NoOps’ (i.e. a model with fully automated cloud-based hardware provisioning and no dedicated operations function). In conjunction, an architecture model has been put forward which is referred to as ‘serverless’ (i.e. ephemeral software containers, provisioned on virtual servers within a cloud-based platform, which are instantly created and destroyed in response to demand for additional capacity for a service or app). For some, this is a dream come true – finally, the work can be done without those ‘pesky’ IT operations people. While the technological aspects of both practices are worth exploring, this extremist view is based upon a limited understanding of what IT operations does - besides procuring hardware and approving changes – and is harmful for both the service provider and their customers. IT operations is much more complex than it might appear at first glance.

There is a thread of the rationale behind the ‘NoOps’ thinking that makes a lot of sense, though – especially when it is transferred to the world of services.

Customers don’t think about services on a daily basis, they care about outcomes. Services are a vehicle to achieve outcomes, and provide a manageable layer between the service provider and the customer where they can agree on the details of what is to be provided, and how. They also hide the more specialised or technical aspects of the service, while retaining transparency and
responsiveness, which are vital for a trusted relationship.

IT operations specialists can be considered to have done a good job of supporting their customer’s outcomes when their presence and the service itself become practically invisible, while remaining approachable should the customer have questions, requests or issues. Removing customers’ headaches is a worthwhile goal. With this in mind, IT operations, as part of the end-to-end value stream, should become a service provider, a platform that provides services to other technical teams. Software development professionals should not have to build/buy and maintain their own tools in support of the CI/CD pipeline – they should be enabled and supported by IT operations. The skills required to support these tools are already there.

Considering the requirements for the separation of duties, asking developers to define their own quality gates would probably not be a good idea. Monitoring tools, relevant for both IT operations and software development, should be designed to be user-customisable. Adding a new report shouldn’t require a filed request, or a ‘ticket’, but should be done by the software development or an IT operations specialist who needs it, with a few clicks in the monitoring environment. Self-service concepts should be used as much as possible.

The hard work behind this falls to IT operations, but building customizable tools is a better approach than processing copious individual request tickets. This is comparable to user-focused services, where a self-service portal can help the user achieve the required results faster, with no added workload for the service provider. The reduction in variation in solutions and platforms in both cases is a bonus.

Platform services can be of great help when seeking to align ITSM and software development, which in turn can help align quality assurance, security, and governance, to provide a holistic service experience for the customer.

8.2 ALIGNING ITSM AND SOFTWARE DEVELOPMENT

We could take this thinking one step further and say these areas should be integrated. This is not to advocate for getting rid of specialization or specialized teams, where models like this are useful. It’s more about very close collaboration, and mutual enablement.

Figure 8.1 below shows a vision of the fully aligned service management approach, with all relevant capabilities recognized and leveraged across the work streams. Platform services are provided by IT operations specialists and are used by both ITSM and software development professionals. Security and quality requirements and practices interact with all parts of the model, and should be built-in where possible, rather than being the responsibility of a separate team that frequently ends up working in isolation.

Figure 8.1 A full view of ITSM and Software Development alignment.
There are many ways to structure teams in a service provider’s organization. No practice should be blindly followed, and no model directly copied without understanding its meaning and intent, to avoid cargo-culting. Some organizations may require a separate DevOps team, which for many in the DevOps community feels like going against the core principles of the DevOps movement, as it has a risk of introducing yet another silo.

Other organizations go straight to matrix models with virtual teams aligned to specific products and/or services. For others, introducing a temporary structure to kick-start the improvement initiative might be the quickest way to achieve meaningful results. And some may keep their existing structures and focus on enabling communication within the existing structures to achieve the best results.

The opportunities are endless.

Afterword

We are interested in your thoughts on the topics explored in this Guidance Paper. Which sections did you find most useful, and how are you planning to use what you have learned? Which sections left you with questions, and requirements more guidance? Were any of the sections confusing and would benefit from an improved explanation?

The DevOps philosophy is well aligned with the intent of ITSM and the mind-set of the end-to-end service view as described in the ITIL framework. The world of ITSM is complex, and many organizations have struggled with adopting the end-to-end service view. This is why this paper has focused on the most widely adopted part of ITSM – IT operations as the entry point. This paper is not meant to be a comprehensive guide to everything DevOps – it’s the first step to understanding the philosophy, and how it fits with how organizations have been working in the past, and in many cases, how they work today.

We acknowledge that leveraging the improvement opportunities DevOps practices bring to ITSM requires more guidance. With your input, we can make sure that the challenges most relevant right now are addressed, and that all the guidance we provide links back to the context of your challenges and opportunities.

Visit www.axelos.com/best-practice-feedback - we would love to hear from you.

Appendix A: A subset of DevOps concepts and practices

<table>
<thead>
<tr>
<th>Concept, practice, or method</th>
<th>Currently biggest impact in ITSM on:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Integration</td>
<td>Change management</td>
</tr>
<tr>
<td>Continuous Delivery</td>
<td>Release and deployment management</td>
</tr>
<tr>
<td>Continuous Deployment</td>
<td>Service asset and configuration management</td>
</tr>
<tr>
<td></td>
<td>Change management</td>
</tr>
<tr>
<td></td>
<td>Release and deployment management</td>
</tr>
<tr>
<td>Infrastructure as Code</td>
<td>Event management</td>
</tr>
<tr>
<td></td>
<td>Incident management</td>
</tr>
<tr>
<td></td>
<td>Capacity management</td>
</tr>
<tr>
<td>Kanban</td>
<td>Microservices</td>
</tr>
<tr>
<td>Serverless computing</td>
<td>Service asset and configuration management</td>
</tr>
<tr>
<td></td>
<td>Release and deployment management</td>
</tr>
<tr>
<td></td>
<td>Capacity management</td>
</tr>
<tr>
<td></td>
<td>Demand management</td>
</tr>
<tr>
<td></td>
<td>Event management</td>
</tr>
<tr>
<td>Simian Army</td>
<td>Incident management</td>
</tr>
<tr>
<td></td>
<td>IT service continuity management</td>
</tr>
<tr>
<td></td>
<td>Availability management</td>
</tr>
<tr>
<td>Test automation</td>
<td>Change management</td>
</tr>
<tr>
<td></td>
<td>Release and deployment management</td>
</tr>
</tbody>
</table>
## Appendix B: ITIL Guiding Principles

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
</table>
| Focus on value        | Everything the service provider does needs to map, directly or indirectly, to value for the customer and/or the organization. This is one of the most fundamental principles of ITIL and ITSM.  
It is the customer who determines what is of value to them, not the service provider. Continual improvement must be focused around making improvements that will result in greater value being delivered to the customer. |
| Design for experience | It is critical to retain the focus not only on business/customer value, but also on the experience that both customers and users have when they interact with the service or service provider. This is frequently called the ‘customer experience’ and it must be actively managed. |
| Start where you are   | Resist the temptation to start from scratch and build something new without considering what is already available to be leveraged. Based on the vision for the future and how that will deliver value to the customer, there is likely to be a great deal in the current services, processes, programmes, projects, people etc. that can be used to create that future. |
| Work holistically     | No service or component stands alone. The results delivered to the organization or customer will suffer unless the service provider works on the whole, not just on the parts.  
Results are delivered to the customer through the effective and efficient management of a complex integration of hardware, software, data, processes, architectures, metrics, tools, people and partners, all coordinated to provide a defined value. |
| Progress iteratively  | Even huge initiatives have to be accomplished iteratively. Resist the temptation to do everything at once. By organizing work into smaller, manageable sections that can be executed and completed in a timely manner, the focus on each smaller improvement is sharper and easier to maintain.  
Improvement iterations can be sequential or simultaneous, based on dependencies or lack thereof. The key is for each individual improvement to be manageable and managed, to ensure that real results are returned in a timely manner and built upon to create more improvement. |
<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observe directly</td>
<td>To know what is really going on, measure and/or observe it directly. Be sure to base decisions on information that is as accurate as it can be. Going to the source allows a reduction in the use of assumptions which, if proved unfounded, can be disastrous to timelines, budgets and the quality of results.</td>
</tr>
<tr>
<td>Be transparent</td>
<td>To know what is really going on, measure and/or observe it directly. Be sure to base decisions on information that is as accurate as it can be. Going to the source allows a reduction in the use of assumptions which, if proved unfounded, can be disastrous to timelines, budgets and the quality of results.</td>
</tr>
<tr>
<td>Collaborate</td>
<td>When the right people are involved in the right ways, improvements benefit from better buy-in, better relevance (because better information is available for decision-making) and better likelihood of long term success.</td>
</tr>
<tr>
<td>Keep it simple</td>
<td>If a process, service, action, metric etc. provides no value or produces no useful outcome, then eliminate it. In a process or procedure, use the minimum number of steps needed to accomplish the objective(s). Although this principle may seem obvious, it is frequently ignored, resulting in overly complex work methods that rarely maximize outcomes or minimize cost.</td>
</tr>
</tbody>
</table>

12 Appendix C: Recommended additional reading

- ITIL Practitioner Guidance (AXELOS, 2016)
- Dominica DeGrandis, Kaimar Karu: Using Kanban in IT Operations (AXELOS, 2016)
- Mark Schwartz: The Art of Business Value (IT Revolution Press, 2016)
- Jez Humble, Joanne Molesky, Barry O’Reilly: Lean Enterprise: How High Performance Organizations Innovate at Scale (O’Reilly Media, 2014)
- Jez Humble, David Farley: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Addison-Wesley Professional, 2010)
- Mike Rother: Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results (McGraw-Hill Education, 2009)
End notes

1 *ITIL Practitioner Guidance*, page 143

2 From http://www.lean.org/WhatsLean/ accessed 25 April 2017


4 https://www.merriam-webster.com/dictionary/scientific%20method
About the author

Kaimar Karu is the Head of Product Strategy and Development at AXELOS, leading a team of international experts that look after the IT Service Management (ITSM) and Project and Programme Management (PPM) best practice portfolios, which include ITIL and PRINCE2 among other products. He has a diverse career background in IT, having worked in IT Operations, Software Development, Project and Programme Management, and IT Service Management. He has a passion for helping people learn and improve, and has worked as a teacher, trainer, and coach in schools, universities, and professional training organizations across Europe. Before joining AXELOS in 2014, Kaimar spent three years immersed in the world of startups, Nordic pragmatism, and multi-billion dollar acquisitions, working with Skype in his native Estonia.

Kaimar has held the position of the president of itSMF Estonia since 2013, holds a master’s degree in Philosophy, and has won a 2nd place in the national beer sommelier competition. He’s now based in London, but is on the road most of the time, exploring good, emergent, and novel practices in organizations around the world. He tweets about good food, best practices, and continual improvement at @kaimarkaru.

Reviewers

We would like to thank the following:

About AXELOS

AXELOS is a joint venture company, created by the Cabinet Office on behalf of Her Majesty’s Government (HMG) in the United Kingdom and Capita plc to run the Global Best Practice portfolio. It boasts an already enviable track record and an unmatched portfolio of products, including ITIL®, PRINCE2® and RESILIA™. RESILIA is the new Cyber Resilience Best Practice portfolio.

Used in the private, public and voluntary sectors in more than 180 countries worldwide, the Global Best Practice products have long been associated with achievement, heightened standards and truly measurable improved quality.

AXELOS has an ambitious programme of investment for developing innovative solutions and stimulating the growth of a vibrant, open international ecosystem of training, consultancy and examination organizations. Developments to the portfolio also include the launch of PRINCE2 Agile®, the ITIL Practitioner qualification and a professional development programme for practitioners, fully aligned with AXELOS Global Best Practice.

Trade marks and statements

AXELOS, the AXELOS logo, the AXELOS swirl logo, ITIL, MoP, M_o_R, MoV, MSP, P3M3, P3O, PRINCE2 and PRINCE2 Agile are registered trade marks of AXELOS Limited. RESILIA is a trade mark of AXELOS Limited. All rights reserved.

© Copyright AXELOS Limited 2017.

Figures 5.1, 5.2, 7.1 and 8.1 are copyright AXELOS

Reuse of any content in this Guidance Paper is permitted solely in accordance with the permission terms at https://www.axelos.com/policies/legal/permissions-use-of-white-papers-and-case-studies.

A copy of these terms can be provided on application to AXELOS at Licensing@AXELOS.com.

Our Guidance Paper series should not be taken as constituting advice of any sort and no liability is accepted for any loss resulting from use of or reliance on its content. While every effort is made to ensure the accuracy and reliability of the information, AXELOS cannot accept responsibility for errors, omissions or inaccuracies. Content, diagrams, logos, and jackets are correct at time of going to press but may be subject to change without notice.

Sourced and published on www.AXELOS.com.