White Paper

Enterprise DevOps

Building a Service Oriented Organization

ascent
atos thought leadership

Atos

We believe that two of the critical indicators of improved IT performance are shorter lead times and increased reliability; DevOps enables improvements along both of these axes simultaneously.

In this paper we will describe why securing these improvements is important for the vast majority of organizations, whether they build software products, provide IT services or outsource their IT requirements to 3rd party providers.

We define what DevOps is and briefly describe the practices that support it before focusing in more detail on two aspects of DevOps that can be particularly challenging for enterprises: creating the right organizational structure and managing the transformation.

For organizational structure we describe why it is necessary to think of teams within an organization as a set of interacting services. Then we look at how Kanban is not just an appropriate method for managing the work of these teams, but also a proven, low-risk method for enabling a DevOps transformation. We conclude with case studies illustrating the results that a successful DevOps implementation can deliver.

For some organizations many of the approaches we describe may already be commonplace, whilst for others they will seem revolutionary. This reality is consistent with our characterization of DevOps as a journey: one on which different organizations have travelled different distances and at different speeds. By providing you with key takeaways and action points we hope that we can help you to take the next steps on your DevOps journey.

Contents

The DevOps imperative

Why IT performance is critical for success in a digital world

Effective development, deployment, operation and maintenance of software are critical factors for business success for the vast majority of organizations, leading some to say that "now every company is a software company"[1]. This is borne out by the level of investment that organizations are making in IT: 95% of all capital projects include an IT component and 50% of all capital spending is technology related[2]. And often IT is driving business success. For example, one leading UK high-street retailer attributed their entire revenue growth purely to an increase in online sales[3].

Responsiveness is no longer optional

Many organizations now operate in business environments that are unpredictable and changing rapidly[4]. One key to success in these environments is short lead times for software development, both to keep up with competitors and to rapidly try out new features and services with customers. To quote Gartner on this topic: "The digital world demands greater enterprise dexterity in creating value from technology."[5]

In practice, many organizations struggle to achieve this "enterprise dexterity". Only 25% of enterprises feel their software deployment and delivery is effective[6] and the perception of the business is often that IT is too slow or is working on the wrong things[7].

For example, after a period of significant growth, Salesforce found that their release cycles had become "long and unpredictable" which led to late feedback on new features[8]. In another example, one team at Microsoft "had the worst reputation for customer service…they had a five-month lead time on change requests and this, along with their backlog of requests, was growing uncontrollably"[9]. In both of these cases the need for change was recognized: unresponsiveness in software development was hurting their business.

Reliability is critical

Reducing lead times for software development is important in unpredictable business environments. But for any organization that depends on software systems for their business success, reliability is also critical: in order for technology to drive revenue and profitability, it has to be operational. This was a significant concern for online marketplace company Etsy in 2008 when "deployments would routinely fail…with no easy way to restart the services or rollback changes"[10]. Just like the teams at Salesforce and Microsoft, they also recognized the need for change though, in their case, it was poor availability, rather than long lead times, that was hurting their bottom line. Like Salesforce and Microsoft, they were also successful in transforming their software development capability. We will return to each of these stories at the end of this paper to understand how each transformation was achieved.

A new way of consuming software services

The necessity for reduced lead times and increased reliability does not only apply to in-house software development. Organizations that buy software services from 3rd party providers will often have exactly the same needs. However the challenges for these organizations are different.

In many cases funding custom development will be replaced altogether by the use of software, platforms and infrastructure that are provided "as a service". In these cases organizations will consume only what they need, when they need it, and pay only for what they use. Doing this enables them to avoid the costs, timescales and risks associated with rebuilding something themselves which is already commercially available. Instead they can focus their resources on development activities where they can add the most value.

[1] http://www.forbes.com/sites/techonomy/2011/11/30/now-every-company-is-a-software-company/
[2] https://blog.newrelic.com/2015/06/24/gene-kim-why-devops-matters/
[3] http://www.telegraph.co.uk/news/shopping-and-consumer-news/11324611/Click-and-collect-overtakes-home-delivery-at-John-Lewis.html
[4] http://ascent.atos.net/devops-or-die/
[5] Bimodal IT: How to Be Digitally Agile Without Making a Mess, Gartner, 2014
[6] http://www.forbes.com/sites/benkerschberg/2015/02/04/why-devops-integration-and-continuous-delivery-hold-the-key-to-enterprise-mobile-app-dev/
[7] 2-Speed IT, Atos White Paper
[8] http://www.slideshare.net/sgreene/the-year-of-living-dangerously-extraordinary-results-for-an-enterprise-agile-revolution-368526/21-Lack_of_visibility_at_all
[9] Kanban: Successful Evolutionary Change for Your Technology Business, David J. Anderson, 2010
[10] http://itrevolution.com/one-of-the-best-devops-talks-on-it-transformation-continuously-deploying-culture-by-rembetsy-and-mcdonnell-velocity-london-2012/
[11] http://ascent.atos.net/a-new-era-of-project-management/

Where a 3rd party is needed for custom software development, supplier selection will need to focus more on the provider's capability to work collaboratively on incremental features which are delivered into production more frequently. Rather than demonstrating that they can manage a large delivery with fixed scope, cost and timescales, suppliers will need to show a willingness to refine and change scope, cost and timescales within an agile framework agreement. Where both sides understand how this relationship can work to mutual benefit, we have seen this approach be highly successful[11].

Organizations that contract in this way will also need to learn new ways to plan and budget for software development. Their role will change from signing-off large batches of work at key milestones (e.g. requirements, functional design and user acceptance) to providing a continuous stream of prioritized requirements and working collaboratively with the supplier to validate the software as it is built.

The urgency for DevOps adoption

As we described earlier, many organizations now rely on software for their business success, and many find themselves operating in rapidly changing business environments. We believe that the extent to which these factors apply to your business can help you understand how urgent DevOps adoption is for you (see **Figure 1**):

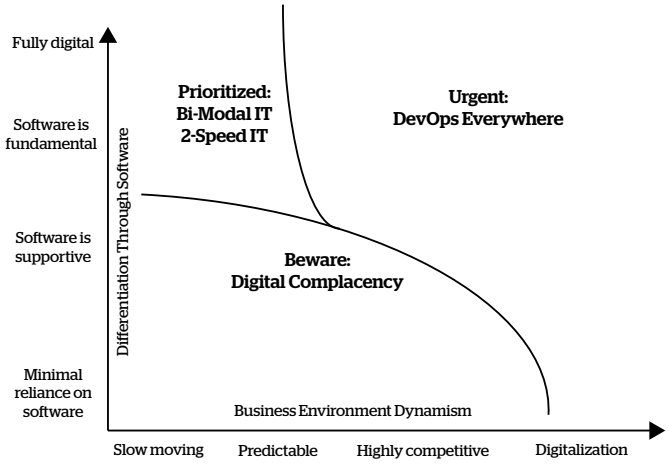

Figure 1 : Urgency for DevOps Adoption

Urgency for DevOps Adoption Model developed by the Atos Scientic Community DevOps track

- If you do not use software at all (or use it only in a supporting role) then you may not need to implement DevOps anywhere. But beware of Digital Complacency: even if you don't see software as a differentiator for you today, it may still be a vital component of your business strategy tomorrow.
- If software is a differentiator for you but you operate in a relatively stable business environment then you may wish to prioritize and target your DevOps implementation using models such as Bi-Modal IT[12] and 2-Speed IT[13] which can help you identify where the return on investment will be greatest.
- If your entire business is built around software (fully Digital) and your business context is undergoing significant disruption (Digitalization) then implementing DevOps across your entire enterprise will be an urgent concern.

To avoid Digital Complacency it is important to consider not only your current position but where you will be in the future and how ready you are to respond to that change. As a cautionary tale consider that in the year 2000 video rental firm Blockbuster enjoyed a dominant market position (and would have been placed somewhere in the bottom left of this diagram). However, 10 years later, they found themselves on the bottom right: a traditional company using software only in a supportive role but now operating in a market that was undergoing massive digital disruption. Unable to respond quickly enough to that change they lost market share, ultimately resulting in them filing for bankruptcy[14].

Our journey from New Norm to DevOps

In 2009 Atos published "Emerging Business Technology – Unleashing Digital Potential"[15] in which we identified how organizational thinking needed to change towards a "New Norm" in order to deliver results in a digital world. Our work applying this thinking with numerous clients across many different markets enabled us to refine these concepts further and led to our publication of "2-Speed IT" in 2014. In that paper we analyzed how the demand for IT has increased over time whilst the capability to meet this demand has declined, and we outlined high-level approaches to closing this gap based on the successful results we had achieved with our customers.

This paper represents the next evolution in our thinking: DevOps as an enabler for business success in a digital world.

[12] Bimodal IT: How to Be Digitally Agile Without Making a Mess, Gartner, 2014
[13] 2-Speed IT, Atos White Paper
[14] http://www.businessinsider.com/how-netflix-bankrupted-and-destroyed-blockbuster-infographic-2011-3
[15] Emerging Business Technology – Unleashing Digital Potential, Atos White Paper

The DevOps journey

DevOps is a philosophy for how to build software

DevOps is not a highly prescriptive set of rules. Instead we define it as a philosophy for how to build and operate software that encourages teams to focus on business value, work collaboratively, deploy software more frequently in smaller increments and build reliable solutions. DevOps promotes continuous improvement across all of these dimensions and, as such, is a journey without a final destination[16].

Business value

Work is managed across the entire end-to-end software value stream (from initial idea through to release and maintenance of features that are meeting the business need). Teams self-organize around the work[17], taking full responsibility for the development of the software and the operation and maintenance of it in production. By collecting metrics over the entire value stream, teams can optimize themselves for the delivery of business value using paradigms such as The Theory of Constraints[18].

Collaboration

DevOps aims to break down silos and remove any "us and them" mentality between different departments. In fact, this is where the term DevOps stems from: the idea that the people developing software and the people deploying and maintaining it in production should collaborate closely together, often in the same team, or even without any role distinction. However this increase in collaboration should not be restricted just to development and operations. We are seeing the emergence of terms like BizDevOps (to emphasize collaboration with the business), CusDevOps (to emphasize collaboration with the customer) and DevSecOps (to emphasize that everyone has responsibility for security). But these new terms can be unnecessarily limiting: DevOps promotes greater collaboration between all stakeholders including (but not restricted to) end users, customers, security experts, compliance managers, sales people and business executives.

Increased collaboration also improves organizational resilience because knowledge is shared across the team (thus reducing dependency on any one individual). One specific technique that supports this is pair programming, where developers work on code in pairs, thereby ensuring quality control through peer review is implemented at the time new code is being written (and avoiding rework). In the case of Extreme Programming all code is written in this way and team members rotate who they work with every day.

Frequent deployment

By promoting more frequent delivery and deployment of software, DevOps enables more business value to be delivered faster[19]. Deploying more frequently enables rapid feedback from end-users about how features are being used and what new features are required, which can then be used to validate and inform future development activities. One example of this approach is Experiment Driven Development supported by A/B testing (where different versions of software are live in production at the same time and usage patterns are monitored). Another example is IBM's Design Thinking[20] which also aims to create a rapid feedback loop between end users and system designers. One positive outcome of these approaches is an end to throw-away proofs-of-concept: rather an initial version is created and then, if it delivers value, continually evolved based on feedback. This avoids costs because you do not need to build the same thing twice (once as a prototype and then again for real usage).

Deployment Pipeline

One technique that is heavily associated with DevOps is having a defined and highly automated method of moving a change from the point at which a developer has finished coding it through to it being in production and available for all users.

To effectively support DevOps the Deployment Pipeline should:

- Be highly automated
- Detect errors early (both through automated testing and monitoring of the software in production)
- Reduce the risk of a deployment causing downtime
- Enable straightforward roll-back in case of issues

[16] https://ascent.atos.net/devops-the-answer-to-everything/
[17] Kanban from the Inside: Understand the Kanban Method, connect it to what you already know, introduce it with impact, Mike Burrows, 2014
[18] The Goal: A Process of Ongoing Improvement, Eliyahu M. Goldratt and Jeff Cox, 2004
[19] http://ascent.atos.net/agile-myths-will-deliver-faster-cheaper/
[20] http://www.ibm.com/design/thinking/

To support frequent delivery and deployment, many activities are extensively automated: testing, code inspection, software build, deployment and monitoring of the software once it is in production. The extensive automation required for successful DevOps is often supported by the use of cloud based tooling and especially IaaS (Infrastructure as a Service) and PaaS (Platform as a Service). For example, in one case we have worked with clients to setup a PaaS with a self-service portal to automatically provision environments for the development, delivery, testing and deployment of applications in order to reduce lead-times.

Reliability

DevOps places a strong emphasis on delivering software that is reliable in the broadest sense: a combination of meeting the business needs and being available. Practices such as Peer Review and Test Driven Development help to ensure that reliable software is developed in the first place. Building Self-Healing Resilient Software (described in **Table 1**) keeps availability as high as possible even if issues arise with the software itself or the underlying infrastructure: some organizations even intentionally create failure scenarios in production on a random basis in order to incentivize teams to build fault-tolerant systems[21].

DevOps Word Cloud by Atos Scientific Community DevOps track

ITIL and DevOps

We do not consider that ITIL and DevOps are contradictory approaches. We see alignment in many areas (for example ITIL's Continual Service Improvement). However, we do recognize that the way ITIL is often implemented in practice means that it can be associated with overly bureaucratic processes (such as every change needing approval at a Change Advisory Board) which are not consistent with the DevOps philosophy.

Practices, technology and tooling

There are a number of practices that support DevOps which have been extensively documented and are well supported by technologies and tools. We believe that some of these can be classed as "Best Practices" for software development: they are widely regarded as the best approach and are practices that organizations should be using now (see **Table 1**). Others we have classed as "Next Practices". These practices are already being used successfully by many organizations and we expect that many will become Best Practices in the next 2-3 years. These are practices that organizations should be evaluating today and planning to use in the future. Evaluating and adopting these practices (and the supporting techniques and tooling) can be achieved through a combination of consultancy, training and coaching. The fundamental approach will be similar whether an organization has 5 employees or 50,000.

Where larger organizations have a unique challenge is how they organize themselves to enable DevOps and apply this at scale. This is the topic we will cover next.

[21] https://insights.sei.cmu.edu/devops/2015/04/devops-case-study-netflix-and-the-chaos-monkey.html

Best Practices

Practice	Description	Supporting techniques (listed alphabetically)	Example supporting technologies/tools (listed alphabetically)	Key DevOps enabler
Continuous Integration	Frequent integration of code into a shared repository which is verified by an automated build.	Branch by Abstraction Build Automation Feature Branch Feature Toggle Test Automation Trunk Based Development (TBD)	Coverity Git Jenkins Maven Nexus Selenium SonarQube	Frequent deployment
Continuous Delivery	Delivering every change to a quality assurance environment and ensuring business applications and services function as expected through rigorous automated testing.	IaaS Infrastructure as code PaaS	Alien4cloud Ansible AWS Azure Chef Cloud Foundry Docker GoCD Kubernetes OpenShift Puppet	Frequent deployment
Test Driven Development	Writing automated tests before writing code.	Refactoring	Mocking frameworks xUnit family	Reliability
Peer Review	Peer review of code and test cases.	Pair Programming Pull Requests	BitBucket Gerrit GitHub Phabricator Review Board	Collaboration

Next Practices

Practice	Description	Supporting techniques (listed alphabetically)	Example supporting technologies/tools (listed alphabetically)	Key DevOps enabler
Continuous Deployment	Automatic deployment into production of changes that pass all automated tests.	Blue/Green Deployment Canary Releases Dark Launching Deployment Pipeline IaaS Infrastructure as code PaaS Push Karma	Alien4cloud Ansible AWS Azure Chef Cloud Foundry Docker Kubernetes OpenShift Puppet	Frequent deployment
Release vs. Deploy	Clearly differentiating between deployment (the technical act of installing software in a given environment) and releasing (the business decision to deliver new features to end users).	Canary Releases Feature Toggle	ff4j togglz	Reliability
Behaviour Driven Development	An extension of Test Driven Development where tests are specified in business language that describes the desired system behaviour.		Cucumber	Business value
Experiment Driven Development	Implementing features as "experiments" to validate business assumptions about user behaviour.	A/B testing Design Thinking Feature Toggle Lean Startup		Business value
Blameless Post-mortem	Holding a blameless post-mortem after each incident in production with the goal of improving the system and limiting the negative impact.	A/B testing Design Thinking Feature Toggle Lean Startup		Reliability
Self-Healing Resilient Software (Anti-Fragile)	Building systems capable of automatic detection of failures and restoration of service without human intervention.	Bulkhead pattern Chaos monkeys Circuit breaker pattern	Alien4Cloud Atos Compose Hystrix Simian Army	Reliability
Auto-Scaling Software	Architecting software so that it can be easily scaled by running multiple instances.	12 Factor App Actor Based Programming Serverless Architecture	Alien4cloud AKKA AWS Azure Cloud Foundry OpenShift	Reliability

Table 1: DevOps best practices and next practices

Developed by the authors in collaboration with other experts (see acknowledgements)

Organizational structure

Choosing a structure that supports DevOps at scale

To successfully implement DevOps, teams should be autonomous and cross-functional: frequent new releases into production will not be possible if every change requires the explicit involvement and approval of several other departments and teams (for example database administrators, system testers and release engineers). These teams must also not be too large: research has shown that team sizes of 5-9 give the best balance of performance in terms of productivity, predictability, responsiveness and quality[22]. Many people describe this ideal team size as the "2 pizza team": a team that can be comfortably fed by two (probably large) pizzas[23].

For small companies (with 10 or fewer employees) this is not a challenge: they are typically structured as one team which is, by default, cross-functional and autonomous. This team will naturally be highly aligned with business goals; the team is the company.

However, once an organization grows to a size where more than 10 engineers are required to develop and maintain the software that the business needs, then these engineers will need to be divided into separate teams to avoid growing beyond the optimum "2 pizza" size. At this point decisions have to be made about how to structure the organization. Here we discuss three approaches to organization structure: functional, autonomous cross-functional, and service oriented. As we will see here, some of these structures are more suited to DevOps than others.

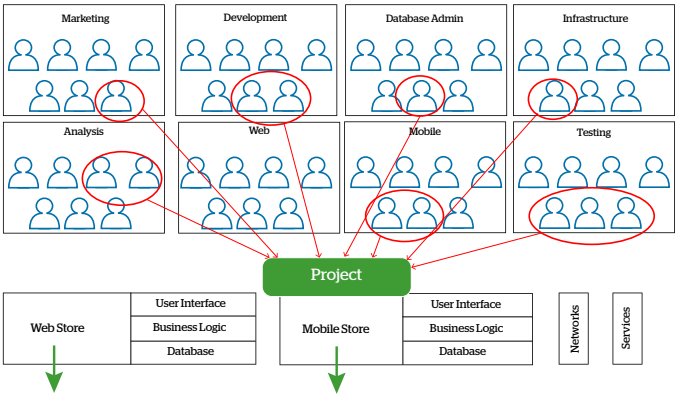

Figure 2: Functional Organization

Functional organization

One traditional approach to structuring an organization (which does not support the DevOps philosophy) is to create teams of people who share certain expertise (for example database administrators, software developers, testers, etc.). This is usually referred to as a functional (or siloed) organization. These teams are not aligned with the software applications that deliver value to the business. When one of these applications needs to be enhanced, a common approach is to commission a project team:

"The team is assigned a project manager, and the project manager then collaborates with various silos to obtain "resources" for each specialty needed to staff the project."[24]

In practice, there are usually many projects running in parallel in a large organization and they often have to compete with each other for these resources. There is a large amount of explicit coordination required, both to ensure that resources are available at the right time, and to ensure that projects do not conflict with each other. Even with perfect planning, resources sometimes have to be diverted without prior notice to address high priority incidents not related to the project they are assigned to. Furthermore, if a project is dependent on one department that happens to be especially busy at one point in time, then the project may be delayed: not because of the time taken to do the work but because of the length of time that the project has to wait for a resource to become available. **Figure 2** shows a highly simplified illustration of functional organization with the flow of value to the business highlighted in green. The software applications deliver value and it is projects that increase this value over time. Because the organizational hierarchy is not aligned with the delivery of value to the business there can be a gap between what the business wants and how people on each team are incentivized (explicitly or implicitly).

Furthermore, with a functional organization like this, the design of the systems will inevitably reflect the specialisms of each department (known as Conway's law[25]). As we will see later, this kind of architecture does not readily support frequent and independent releases of individual software components.

[22] Impact of Agile Quantified LKUK 2014, Larry Maccherone, Presented at LKUK 2014, https://lkuk14.sched.org/event/1rzAA6V/the-impact-of-lean-and-agile-quantified-2014
[23] Succeeding with Agile: Software Development Using Scrum, Mike Cohn, 2009
[24] Migrating to Cloud Native Application Architectures, Matt Stine, 2015, https://download3.vmware.com/vmworld/2015/downloads/oreilly-cloud-native-archx.pdf
[25] How do committees invent? By Melvin E. Conway, 1968, http://www.melconway.com/Home/Committees_Paper.html

Autonomous cross-functional teams

An alternative to functional organization is cross functional organization. In this arrangement teams are aligned with the software applications that deliver value to the business and are accountable for the successful delivery of that value. They have a high degree of autonomy and the full complement of skills required to develop, deploy, operate and maintain their application. They are naturally focused on delivering business value as the management hierarchy and the flow of value is aligned.

It has been well established in the field of product development that autonomous cross-functional teams are the optimal choice for reducing lead times[26]. Part of this reduction in lead time is because of "the willingness of team members to work outside of their primary specialty". For example, more people on a team might perform testing or release activities if that is what is required at a given point in time (which means that those activities do not become a bottleneck).

One objection to this approach is that it is less efficient than organizing teams by function. However, modern management thinking has learned lessons in this area from large military organizations:

"Routine tasks were standardized and everyone was trained in how to do them. Today they are called standard operating procedures or SOPs. They are very useful because they create uniformity and therefore predictability where that has a high value. They enhance efficiency by enabling these tasks to be carried out at speed with little supervision."[27]

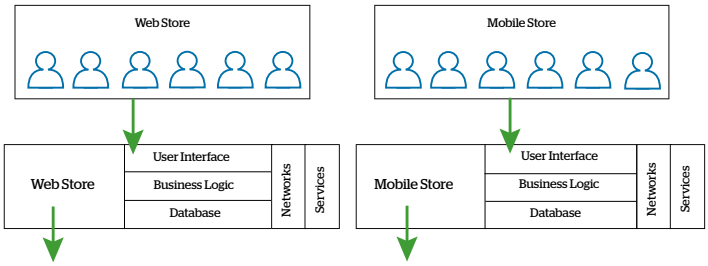

Figure 3: Autonomous Cross-Functional Teams

The Best Practices and Next Practices that we have described earlier are the Standard Operating Procedures for DevOps and they enable the use of cross-functional teams to increase the pace of delivery whilst also maintaining high efficiency.

However there are still two key challenges with this approach which relate to enabling reuse between different applications and scaling the number of people who can work on an application.

With this approach to organization, reuse between teams is not encouraged: in **Figure 3** the business logic for the Web Store and the Mobile Store may be very similar, but to retain their independence and autonomy, the teams may choose not to share the code base.

Scalability poses two issues. Firstly, even for medium sized applications, it can be difficult to put together a single team that has all the required skills and expertise to develop and deploy the complete application. Secondly, some applications are so large that they will require more than one team to develop and maintain them.

[26] Managing the Design Factory: A Product Developers Tool Kit, Donald Reinertsen, 1998

[27] The Art of Action: How Leaders Close the Gaps Between Plans, Actions and Results, Stephen Bungay, 2010

Feature teams and feature slicing

One way of addressing some of the scalability challenges is for more than one autonomous cross-functional team to work on the same software application in parallel. They work across the entire application stack to deliver specific features (hence they are known as feature teams). To facilitate this, work must be divided into features that can be independently developed (known as feature slicing).

Each team must be able to deliver new features independently of each other. To be able to release independently of other teams, feature branches may be created in the software version control system. However if these feature branches are long lived, this can lead to painful merges[28]. An alternative approach (which avoids this problem) is to break down large changes into multiple smaller changes, each of which can be committed to the main source repository and deployed independently (known as Trunk Based Development). One useful technique which supports Trunk Based Development is Branch by Abstraction which enables larger changes to be introduced progressively as a series of smaller enhancements .

Using feature teams and feature slicing does support DevOps and can be applied successfully at scale (for example over 600 developers working on a code base of over 10 million lines of code[30]). However, using feature teams and feature slicing do not inherently facilitate reuse between teams working on different applications, nor do they reduce the difficulty of creating teams with the full complement of expertise necessary to work across the entire application.

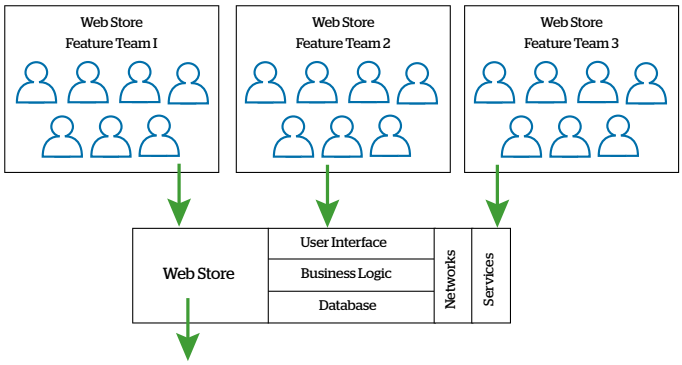

Figure 4: Feature Teams

The service oriented organization

We are increasingly seeing that applications can be architected as a set of microservices (independently deployable, scalable and business-centric software services). By using this architectural style it becomes possible to structure the organization around these microservices. As shown in **Figure 5**, all the teams are autonomous and cross-functional, and fully responsible for delivering value (either directly to the business or to other internal teams). Requirements flow in the reverse direction to value: if an enhancement to the Mobile Store requires an enhancement to the Shopping Cart, then the Mobile Store team will define the requirements and the Shopping Cart team will build and deploy the updated software that meets them.

Using this approach it is possible to reuse components (for example the Shopping Cart is reused by both the Web Store and Mobile Store applications). Furthermore, whilst each team still needs the skills to develop and deploy across the entire stack for their microservice, the breadth of skills required is usefully constrained by the scope of the microservice. Two additional benefits of this organization relate to team stability and the way that interactions between teams can be managed and optimized by clearly defining how the teams interface with each other.

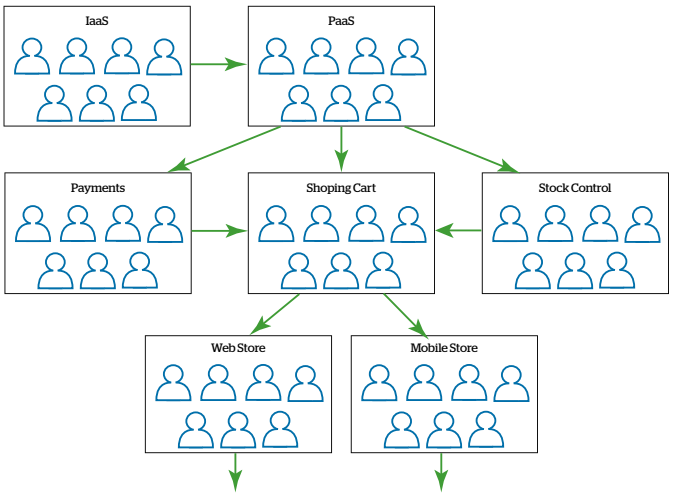

Figure 5: Service Oriented Organization

[28] http://martinfowler.com/bliki/FeatureBranch.html

Team stability

Teams organized around the delivery of microservices will tend to be more stable because, although sometimes new software is created, most development work concerns existing software[31]. In contrast to creating short-lived temporary teams to deliver projects (as described earlier), the life-cycle of a DevOps team will match the life-cycle of the software that they develop and maintain. This improves knowledge retention and boosts productivity: research has shown that teams with greater stability achieve up to 60% better productivity and 40% better predictability, with the best results achieved by teams that have been together for 3 years or longer[32].

Although team stability brings benefits, it can be unsettling for some. For a team that is responsible for a microservice, their work is not "complete" or "finished" until their microservice is no longer in use.

Defined interfaces

Drawing on lessons learned from systems theory and applied to product development, it is recognized that one of the strongest levers for influencing overall organizational performance is the interfaces between teams rather than the teams themselves[33]. These interfaces are far more than a definition of how microservices will interface technically. They should also define how responsive the team can be (using lead-time and throughput metrics based on past performance), what pre-requisites they require in order to start work, how progress is monitored and what the criteria are for work being considered complete. Additionally they will define how work is allocated to the team (and by whom) as well as specifying whose agreement is required to expedite high priority work items. Publishing these team "interface specifications" enables teams to manage their dependencies on other teams and, as such, they are a key to scaling DevOps to be enterprise wide.

Example Team Interface	
Pre-requisites	User story Key acceptance criteria Approved by product owner
Expediting	Requires agreement of product owner and business division manager
Lead time	80th centile: 4 weeks 90th centile: 6 weeks 95th centile: 10 weeks
Throughput	20 user stories per month
Progress tracking	Kanban board Email when status changes
Definition of done	Live in production for 1 week Validated and accepted by requestor

Example Team Interface developed by the authors to illustrate the types of information that should typically be included

[29] http://paulhammant.com/blog/branch_by_abstraction.html
[30] Development and Deployment at Facebook, Dror Feitelson, Eitan Frachtenberg & Kent Beck, 2013, https://research.facebook.com/publications/development-and-deployment-at-facebook/
[31] http://www.infoq.com/articles/kelly-beyond-projects
[32] Impact of Agile Quantified LKUK 2014, Larry Maccherone, Presented at LKUK 2014, https://lkuk14.sched.org/event/1rzAA6V/the-impact-of-lean-and-agile-quantified-2014
[33] Managing the Design Factory: A Product Developers Tool Kit, Donald Reinertsen, 1998

Organizational transformation

Using Kanban to evolve collaboratively towards DevOps

Earlier we described Best and Next Practices (together with examples of the associated technology and tooling) which can support the DevOps philosophy. However, it is estimated that only 30% of the effort required to implement DevOps relates to changes in this area[34]. The remaining 70% of the implementation effort is needed to introduce the necessary processes, culture and organizational structure.

The challenge of this organizational change should not be underestimated: numerous research studies over the last two decades have concluded that only 30% of transformation programs are successful[35]. However, in recent years we have seen Kanban emerge as a lower-risk method for driving organizational change which is highly aligned with DevOps. Our practical experience with Kanban has confirmed that it is effective in a variety of different contexts and can scale to software deliveries involving hundreds of employees.

It is possible for DevOps adoption to take place in a bottom-up manner, driven by employees who understand the philosophy and can foresee the benefits. However, our experience shows that this kind of adoption will usually only occur within small "pockets" of the organization. Furthermore their efforts can be undermined if the stakeholders (whom the team interfaces with) are not sufficiently engaged with the change. Therefore we feel that a vital ingredient for successful enterprise-wide DevOps adoption is executive sponsorship, which includes a recognition that it is not just a technical change for engineers, but a cultural shift that will impact everyone.

There is no longer a target state

In simple terms, traditional approaches to transformation start by identifying the "as is" state and the "desired" state and then manage the transition between these two states as a project. This approach to transformation is fundamentally at odds with DevOps which, as we have described earlier, is a journey without a final destination. What's more, the notion of a fixed pre-defined "desired" state does not make sense in the rapidly changing business environments in which many companies now operate. In these contexts, the correct "desired state" in the future is simply unknowable months or years in advance. What is needed instead is an approach to change which builds organizational resilience by enabling it to adapt and respond to external pressures and demands; a shift from "As-Is and To-Be grand designs" to "Next state, evolving, adaptive change"[36].

Kanban for evolutionary change

Although Kanban can be used to describe a number of different techniques and approaches (for example many Scrum teams will describe their task board as a "Kanban Board"), here we are specifically considering the Kanban Method as first described by David J Anderson in his book "Kanban: Successful Evolutionary Change for Your Technology Business" which was published in 2010. Since its publication, the popularity of this approach has grown and many case-studies have reported successful results through its application[37]. Within Atos, we consider it to be the most effective catalyst for driving enhancements in throughput and quality by embedding a culture of continuous improvement.

Starting with what you do now

The Kanban approach is based on the principle that people within an organization can and will find the best ways of working, provided that goals are clearly articulated to them, they are given information that helps them understand how they work and they have the autonomy to change the way they work. To achieve this Kanban defines nine values, six foundational principles and six core practices[38] which we believe support the philosophy of DevOps and can catalyze the required organizational changes (see **Table 2**).

Kanban explicitly states that you should "start with what you do now" and initially "respect current processes, roles, responsibilities and job titles" focusing instead on organizing the work and letting "people self-organize around it, allowing the system to change as a result"[39]. This side-steps one common barrier to successful transformation: resistance to change because it challenges people's roles and positions within the business. As teams organize around the work and seek to continuously improve, processes and roles may be adjusted if and when required.

Kanban does not seek to impose any specific software development practices or processes either. If a team is using Scrum then they will continue to do so, with Kanban providing a mechanism to continuously improve the way they work (an approach known as Scrumban). We have used Kanban in this way to drive improvements where a team was already practicing Scrum to deliver a major high security government project. In this case we saw cycle times and throughput improve by over 20%. In fact, it often makes sense to use Agile methods like Scrum or XP as the initial process when setting up a new team, with Kanban then providing a means for the team to continuously self-optimize.

[34] Report of the Syntec DevOps Camp-Paris Feb. 2016
[35] http://www.mckinsey.com/business-functions/organization/our-insights/the-irrational-side-of-change-management
[36] Emerging Business Technology – Unleashing Digital Potential, Atos White Paper
[37] http://leankanban.com/case-studies/
[38] Essential Kanban Condensed, David J Anderson and Andy Carmichael, 2016
[39] Kanban from the Inside: Understand the Kanban Method, connect it to what you already know, introduce it with impact, Mike Burrows, 2014

Because Kanban is non-prescriptive, it enables and encourages teams to be autonomous, asking only that they focus on continually improving their ability to deliver business value.

Because Kanban respects existing processes and roles, and emphasizes evolutionary change, we have seen that it can be implemented quickly, with relatively little effort and without meeting significant resistance. However the implementation still needs to be managed so that all stakeholders fully understand what the benefits will be (both for the organization and for them personally) and what will be expected of them. To achieve this, organizations can use an approach known as Systems Thinking Approach To Implementing Kanban (STATIK). This is a 9 step process that covers the identification and understanding of services, the design of appropriate Kanban systems and the socialization of those systems within the business.

Perhaps surprisingly for a method that emphasizes changing so little at the outset, we have seen many examples of it delivering radical transformation in ways of working over short timescales.

Visualizing work

One challenge with managing software work (in common with all knowledge work) is that the end-product is not physical. In a factory you can literally see what people are working on and you can identify where there are bottlenecks by observing where inventory is accumulating. In contrast, when you watch a team of software developers at work you cannot see what they are working on, where the bottlenecks are, what work is blocked or how each person is contributing to the end goal. This makes it hard for teams to self-organize around the work and optimize for faster lead times.

Kanban addresses this by visualizing the work. Usually the preferred approach is to use a physical board that can be easily seen by the whole team all of the time. The board is divided into separate columns: one for each knowledge discovery step that work has to pass through (for example analysis, design, code, test, integration and deployment). Sticky post-it notes or magnetic cards are used to represent each software change and are placed in the column that reflects their current status. Other things that are commonly visualized on the Kanban board are the priority of the change, who is currently working on it and whether it is blocked. Software tools can be used as a complement to or instead of physical boards to work around constraints (such as the team not being co-located) and to automate the collection of metrics.

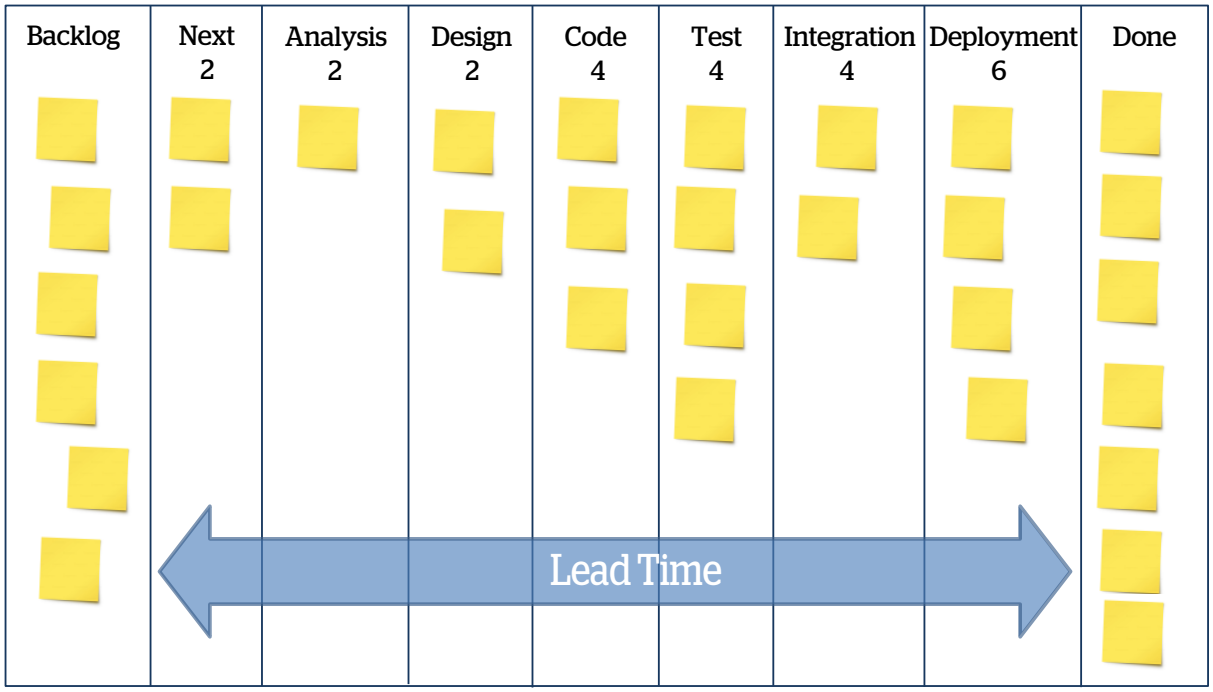

Figure 6: Example Kanban Board

Explicit policies are defined (and usually displayed on the Kanban board) which state what the pre-requisites are for a change to move from one column to the next (often called "definitions of done"). They should include all requirements, including non-functional requirements such as security, performance, accessibility and code quality. These policies provide clarity, not only for the team(s) doing the work, but also to those outside of the team: they define how the team interfaces with the wider organization. Adjusting these policies is one of the ways that teams can seek to improve the process.

We have found that Kanban boards not only help teams to manage their work, but also act as an excellent information radiator to keep interested stakeholders updated of progress across distributed sites.

Kanban			DevOps Philosophy				
Values	**Foundational Principles**	**Core Practices**	**Business value**	**Collaboration**	**Frequent deployment**	**Reliability**	**Continuous improvement**
Balance	Manage the work	Limit work in progress			●	●	●
Transparency	Service orientation	Visualize		●			●
		Make policies explicit		●			●
		Implement feedback loops	●	●			●
Collaboration		Improve collaboratively, evolve experimentally		●			●
Customer Focus	Focus on customer needs	Manage flow	●		●		●
Flow							
Leadership	Encourage acts of leadership at all levels			●			●
Respect	Start with what you do now		●	●			●
Agreement	Agree to pursue evolutionary change		●	●			●

Table 2: How Kanban Supports DevOps

Kanban to DevOps mapping developed by the Atos Scientific Community DevOps track

Limiting work in progress

Kanban implements Work In Progress (WIP) limits. These are commonly visualized by writing a number at the top of each column on the Kanban board which is the maximum number of changes that will be allowed to accumulate in that column at any one time. Research has shown that lowering the overall WIP results in shorter lead times and improved reliability (a 75% reduction in defects)[40] .

Limiting WIP also creates slack in the system. Kanban recognizes the value of slack as it "is still needed: The whole quote should be exactly: "can be used to improve responsiveness to urgent requests and to provide bandwidth to enable process improvement…without slack there is no tactical agility in the business"[41].

Controlling WIP also makes it easier to spot issues and bottlenecks. Without WIP limits, if a work item becomes blocked it is often easier for team members to start work on something else in the backlog rather than un-blocking the change that has already been started (a popular Kanban mantra is "stop starting and start finishing"). Without WIP limits it is easy for everyone to be "busy" which can hide where the real bottleneck is. Some have described the practice of reducing WIP limits as lowering the water level in the value stream so that "even the smallest problems in our system will be visible, just as the rock in a canal will be visible"[42].

Feedback loops

To enable continuous improvement, Kanban defines a number of regular review meetings at all levels within the organization, ranging from daily "stand-ups" (usually around the Kanban board) through to strategy reviews where the direction for teams across the entire enterprise is set[43].

Metrics

Kanban encourages the collection and analysis of metrics to inform and validate process improvement activities and enable better planning of work. Two key metrics are throughput (the number of changes delivered within a given time period) and lead time (the elapsed time that it takes for a change to move through the WIP limited part of the system). These metrics allow forecasts to be made for how long it will take to deliver future work based on past performance. Statistical methods can be used to build forecasts with specific confidence levels (for example a 90% confidence that a set of tasks can be completed within 8 weeks). The enterprise can use this data to plan work effectively using risk-based approaches to scheduling such as Cost of Delay[44] and Real Options[45].

A catalyst for organizational change

Kanban does not attempt to redefine the current organizational structure. However we have found that, in practice, teams will often reorganize themselves when given the insight into the overall value stream that Kanban provides and asked to optimize for lead time and throughput. For example, in one case, deployment was a bottleneck and developers were quite willing to take on deployment activities when they could see that this was what was constraining delivery lead times. In another case, developers, an architect and a test manager all picked up testing tasks to address a testing bottleneck, enabling the whole team to stay on track. We have also seen cases where team members volunteered to learn new technologies because they could see how it would alleviate a bottleneck within the system. Whilst not mandating cross-functional teams, Kanban naturally leads to this by aligning people around the delivery of business value.

In summary, we have found that Kanban offers an effective, low-risk approach to organizational change which is highly aligned with DevOps. Kanban can be scaled across an enterprise and enables you to manage your organization as a set of interconnected services.

[40] Impact of Agile Quantified LKUK 2014, Larry Maccherone, Presented at LKUK 2014, https://lkuk14.sched.org/event/1rzAA6V/the-impact-of-lean-and-agile-quantified-2014
[41] Kanban: Successful Evolutionary Change for Your Technology Business, David J. Anderson, 2010
[42] https://hakanforss.wordpress.com/2014/08/18/how-to-improve-flow-efficiency-with-scrum-agile2014-qa/
[43] Essential Kanban Condensed, David J Anderson and Andy Carmichael, 2016
[44] The Principles of Product Development Flow, Donald Reinertsen, 2009
[45] Commitment: Novel about Managing Project Risk, Olav Maassen, Chris Matts & Chris Geary, 2013

Conclusion

Taking the next steps on your DevOps journey

In this paper we have defined DevOps as a philosophy for how to build and operate software which encourages teams to focus on business value, work collaboratively, deliver software more frequently in smaller increments and build reliable solutions. We have explained that DevOps is associated with several practices which are supported by a range of techniques, tools and technologies. Enterprise DevOps raises two particular challenges: creating the right organizational structure, and managing the transformation.

For each of these we have discussed approaches that have been proven to work, both by us in collaboration with our clients and also widely in the technology industry. At the start of this paper we described specific challenges that were faced by Salesforce, Microsoft and Etsy; let's now return to their stories and see how they successfully overcame them.

Salesforce were concerned that their release cycles had become long and unpredictable. To address this they embarked on an enterprise-wide Agile transformation impacting over 200 engineers and lasting 3 months. This was followed by another 12 months of continuous improvement. The keys to success that they identified were executive commitment, a focus on principles over mechanics, extensive automation and radical transparency. In other words this was DevOps in action, and it delivered impressive results including increased customer satisfaction, higher productivity, higher morale, an increase in on-time delivery and a reduction in mean time to release[46].

The team we described at Microsoft had a bad reputation for customer service due to their long lead times and their large and growing backlog of requests. They implemented a Kanban system which, over a period of 13 months, resulted in a 200% improvement in productivity and a 90% reduction in lead times (as well as completely eliminating their backlog)[47]. What is perhaps most striking in this case study is that there was no radical change; rather this was an implementation of a Kanban system to limit work in progress and drive evolutionary improvement which yielded impressive results.

As we stated earlier, the key driver for change at Etsy was improving reliability rather than lead times. Over the course of three years they radically changed their approach to building, deploying and operating software. They created slack time for improvement projects, reduced batch sizes and prioritized improving "how they worked" over "doing the work"[48]. The result of this was that they deployed into production 6,419 times in 2012 and those deployments were performed by 196 different people. They reported reduced downtime as a result because, although they were deploying more frequently, the likelihood and severity of problems was reduced and the speed with which they could rectify them was faster[49].

What we clearly see from these three examples, and which is also supported by numerous other case studies and our own experience, is that the application of DevOps can radically transform the IT performance of an organization. What's more, DevOps is not an approach that is reserved for small start-ups or businesses that were born-in-the-internet. By using the Best and Next Practices associated with DevOps and structuring your teams as we have described, it can be applied at scale in any enterprise. Furthermore Kanban provides a low risk way to implement a DevOps philosophy and drive the required change in culture and mindset. DevOps may be a journey without a final destination, but it is also a journey on which any enterprise can take steps today, and realize significant benefits tomorrow.

Key Takeaways	
Urgency	Evaluate the urgency of DevOps adoption based on your business context and consider using models like Bi-Modal IT or 2-Speed IT if a prioritized approach makes sense for your organization.
Consuming IT	Assess how you consume IT services and consider where you can use SaaS, PaaS or IaaS solutions. Where you outsource software development think about how your relationship with providers needs to evolve.
Best Practices	Establish which Best Practices you are already using and where additional training, consultancy and coaching may be required.
Next Practices	Evaluate DevOps Next Practices to understand which you should consider adopting within your organization and over what timescales.
Organizational Structure	Understand the extent to which your existing organizational structure is appropriate for DevOps and where the key challenges are.
Evolutionary Change	Implement Kanban as a low risk method to catalyze your DevOps transformation.

[46] http://www.slideshare.net/sgreene/the-year-of-living-dangerously-extraordinary-results-for-an-enterprise-agile-revolution-368526/

[47] Kanban: Successful Evolutionary Change for Your Technology Business, David J. Anderson, 2010

[48] http://itrevolution.com/one-of-the-best-devops-talks-on-it-transformation-continuously-deploying-culture-by-rembetsy-and-mcdonnell-velocity-london-2012/

[49] http://www.slideshare.net/beamrider9/continuous-deployment-at-etsy-a-tale-of-two-approaches

Acknowledgements

About the Atos Scientific Community

Publically launched by Thierry Breton, Chairman and CEO of Atos, and sponsored by Hubert Tardieu, the Scientific Community has 125 members from all geographies where Atos operates, representing a rich mix of skills and backgrounds. Its aim is to help Atos anticipate and craft its vision of upcoming technology disruptions and the future business challenges that will be faced by the markets it serves. By making this vision available to its clients, and by investing in areas related to the findings, Atos intends to help its clients make informed decisions regarding the future of their Business Technology solutions.

About the C-LAB & the SICP

Founded in 1985, the Cooperative Computing & Communication Laboratory (C-LAB) is a joint research and development laboratory operated by Atos and the Paderborn University. C-LAB's vision is based on the fundamental premise that the gargantuan challenges thrown up by the transition to a future information society can only be met through global cooperation and intensive linking of theory and practice. This is why, under one roof, staff from university and from industry cooperate closely on joint projects within a common research and development organization together with international partners. In doing so, C-LAB concentrates on those innovative subject areas in which cooperation is expected to bear particular fruit for the partners and their general well-being.

The cooperation of C-Lab has been intensified since 2014 in the Software Innovation Campus Paderborn (SICP). The SICP is a cooperation between ten leading-edge technology companies, the Paderborn University and the Fraunhofer IEM. Thus, the campus continues the success story of the 30-year ongoing cooperation model of the C-Lab.

About the Authors

The authors would like to thank the following members of the Atos Scientific Community for their review of early drafts of this paper: David Cunningham, John Hall, Jan Krans, Luis Lancos, Mischa van Oijen, Rob Price, Purshottam Purswani and Mike Smith.

The authors would also like to thank the following contributors to **Table 1**: DevOps Best Practices and Next Practices: Chris Baynham-Hughes, Mike Burrows, Thierry Caminel, David Cunningham, Adrian Hepworth and Jean-Marc Meessen.

Jean-Marc Cadudal Cloud Solution Architect and Member of the Scientific Community, France Email: jean-marc.cadudal@atos.net Twitter ID: @JMCadudal	**Adam Jackson** Head of Atos Decision Factory and Member of the Scientific Community, UK Email: adam.jackson@atos.net Twitter ID: @adamecjackson	**Hannu Ojasalo** Head of Professional Services and Member of the Scientific Community, Finland Email: hannu.ojasalo@atos.net Twitter ID: @Ojasalo
David Daly (Editor-in-Chief) Global Deal Assurance Manager for Worldline and Member of the Scientific Community, UK Email: david.daly@worldline.com Twitter ID: @DavidDalyWL Host of #DevOpsChat Twitter Chat via @DevOpsStars	**Jean-François James**, Software Architect, Head of French Expert Network for Worldline and Member of the Scientific Community, France Email: jean-francois.james@worldline.com Twitter ID: @jefrajames	**Martin Pfeil** CTO Global Siemens Account and Member of the Scientific Community, Germany Email: martin.pfeil@atos.net Twitter ID: @marpfeil
Prof. Gregor Engels Executive Director C-LAB at Paderborn University, Germany Email: engels@upb.de	**Dr. Simon Oberthür** R&D Manager Mobile & Cloud Systems, Software Innovation Campus Paderborn at Paderborn University, Germany Email: oberthuer@sicp.de Twitter ID: @zottel	**Dick van der Sar** Thought Leader DevOps, The Netherlands Email: dick.vandersar@atos.net Twitter ID: @Dvdsar

About Atos

Atos SE (Societas Europaea) is a leader in digital services with pro forma annual revenue of circa € 12 billion and circa 100,000 employees in 72 countries. Serving a global client base, the Group provides Consulting & Systems Integration services, Managed Services & BPO, Cloud operations, Big Data & Cyber-security solutions, as well as transactional services through Worldline, the European leader in the payments and transactional services industry. With its deep technology expertise and industry knowledge, the Group works with clients across different business sectors: Defense, Financial Services, Health, Manufacturing, Media, Utilities, Public sector, Retail, Telecommunications, and Transportation.

Atos is focused on business technology that powers progress and helps organizations to create their firm of the future. The Group is the Worldwide Information Technology Partner for the Olympic & Paralympic Games and is listed on the Euronext Paris market. Atos operates under the brands Atos, Atos Consulting, Atos Worldgrid, Bull, Canopy, Unify and Worldline.

Find out more about us
atos.net
ascent.atos.net

Let's start a discussion together

CT_161027_SE_DevOps_Building_a_Service_Oriented_Organization-V01