bjss

Enterprise DevOps

ALIGNING DEVELOPMENT AND OPERATIONS

A BJSS WHITE PAPER

Contents

This White Paper is for IT decision makers within CIO/ CTO functions, delivery programmes and Operations. It aims to explain what DevOps is, why it should be adopted, what the main barriers to implementation are, and how to overcome them.

Executive Summary

In an increasingly digital world, IT-enabled innovation is becoming a major competitive differentiator for almost all types and sizes of organisation. Agility in delivering IT systems and the ability to run them reliably and cost-effectively is ever more important.

Many enterprises have made good progress in adopting Agile delivery techniques over the last decade. However, we have observed that this progress frequently does not extend beyond the delivery project itself. Development and testing will be carried out in short iterations, but will be deployed into production much less frequently due to operational acceptance processes. The results of this are:

- Delays in realising business benefits
- Complex 'major releases' that carry high levels of technical and business risk
- Infrequent interaction between delivery and operational teams, hindering development of an effective working relationship

In the first edition of our book BJSS Enterprise Agile, published in 2008, we placed strong emphasis on involving Operations teams early in order to achieve a 'No surprises end-game', and stressed the importance of defining non-functional requirements up front, risk-first delivery planning, deployment automation, incremental technical testing and incremental operational acceptance.

DevOps is a relatively new industry term – coined in 2008 and increasingly gaining traction – that draws on many of the same ideas. It is used to describe both a culture and set of related techniques. A DevOps approach aims to break down the barriers that often exist between Development and Operations teams, enabling them to work together to deliver systems into production reliably, safely and rapidly – and to operate and support them more effectively. An effective DevOps culture:

- Delivers systems to the business faster
- Reduces risks of production changes through non-functional testing
- Supports the objectives of Operations – including those expressed in the commonly used ITIL framework for service management – through automation
- Improves visibility and understanding of all layers of the production environment stack within both Development and Operations teams, helping them prevent and resolve production issues

This White Paper is for IT decision makers within CIO/ CTO functions, delivery programmes and Operations. It explains what DevOps is, why it should be adopted, what the main barriers to implementation are and how to overcome them. Delivery functions should read it in conjunction with the 'Operations' chapter of the 2012 edition of the Enterprise Agile book (http://bjss.co/ea), which provides more detailed advice on how to introduce a DevOps approach within a project.

The first section of this White Paper:

- Explains why there is often friction between Development and Operations teams
- Introduces and defines DevOps
- Explains how it relates to Agile Development, Continuous Delivery and standard Operations teams processes (usually derived from ITIL)
- Shows how DevOps can be enabled by Cloud technology, and in return is essential for delivery of Cloud's promises of cost savings and flexibility

DevOps had its origins in relatively new organisations such as Google and Amazon and has had considerable success in this type of environment. Enterprise IT organisations could also benefit, but do not have the luxury of starting from scratch. As a result, the primary barriers to the adoption of DevOps in the enterprise have little to do with technology. They have everything to do with the difficulty of managing the necessary organisational change, and gaining access to the scarce skills required.

In order to help overcome these barriers, the second section of this White Paper presents five models that can be used individually or in combination as a framework to introduce a DevOps approach:

1. Build out from the Operations team
2. Set up a DevOps skunkworks
3. Build out from the Development team
4. Green field implementation
5. Force DevOps onto Operations

It provides advice on the pros and cons of each, which types of organisation each is best suited for, and how best to implement them.

By using these models to help manage a transition to DevOps, IT organisations can deliver better systems to the business faster, more flexibly and more reliably – which will ultimately result in a more efficient, competitive and successful business.

Enterprise DevOps

ALIGNING DEVELOPMENT AND OPERATIONS

Development vs. Operations

In today's enterprise, Operations and Development teams are often quite siloed. They may work in different buildings, or sometimes different continents. They are often in the office at different times of day, since operations teams on critical 24x7 systems work in shift patterns. The organisation structures are usually distinct. They speak different technical languages, and movement between teams that would help foster a common understanding is rare.

Most importantly, they have separate concerns and goals, and these can often be in conflict.

The more mission-critical a system will be to an enterprise, the more weight the Operations view will generally carry. In high-criticality environments, operational acceptance is an important and often formal process.

The natural tension between the groups has been exacerbated by adoption of Agile methods by development projects. Delivery team requests for increased release frequency have put pressure on traditional acceptance processes. Operations teams often struggle to cope with the increased workload, and sometimes feel that they are steamrollered into accepting systems into production that shouldn't be released. Development teams get frustrated by the resulting delays.

The net result of this conflicting dynamic has been termed the 'wall of confusion' by some DevOps commentators. Teams distrust each other and are unable to understand or sympathise with each other's perspective. Project teams view Operations as Luddites who cannot deliver urgently required development and test environments in time, veto the use of any interesting technology, insert apparently meaningless hurdles in the way of delivery, and cannot automate even the simplest of tasks.

IT Operations' unending fire fighting has typically resulted in it being run like a cottage industry rather than an efficient production line. There have been pockets of automation – for example few enterprises would now manually load and unload tapes, almost all can automate common operating system builds, many have an increasingly virtualised infrastructure that simplifies and reduces hardware procurement and setup effort, and some even have an accurate configuration management database.

However, most enterprise software systems are still deployed and run in a highly manual way. This is partly because they are often built on sprawling architectures consisting of a mishmash of third-party software products and internally developed software of varying vintage and quality that does not lend itself readily to automation.

Primary Delivery concerns	Primary Operations concerns
Project spend.	Lifetime total cost of ownership
Planning in advance	Responding to production incidents
Hitting short term milestones	Doing the right thing for the long term
Minimal formality and maximum flexibility	Defined processes and continuous improvement
Non-production environments	Production environments
High performance for the application	High utilisation to maximise return on infrastructure investment
Meeting the business sponsor's functional requirements, which are easiest for them to assess	Non-functional requirements: predictable performance, reliability, resilience, security, proactive monitoring, diagnostic tools, well documented procedures
Implementing major change quickly	Minimising risks to production systems
Using the latest technology	Standardisation on proven technology.

What is DevOps?

The term DevOps was first coined in 2008. It suffers from similar problems to other industry buzzwords such as Cloud and Big Data in that it is often used in an undefined or ambiguous way.

When we talk about DevOps, we are referring to a culture rather than any specific technique. DevOps aims to break down the barriers that often exist between Development and Operations teams, enabling them to work together to deliver systems into production reliably, safely and rapidly – and to operate and support them more effectively. It aims to develop a collaborative working relationship and foster adoption of a common set of aims and objectives to:

- Deliver IT services that provide strategic value to the business
- Expedite project delivery of new services, reducing project costs
- Deliver services that satisfy both functional and non-functional requirements
- Reduce recurring operational costs through efficient utilisation of infrastructure investments and standards
- Reduce operational service risks associated with changes introduced by projects
- Manage configuration effectively and accurately
- Improve visibility and understanding of the operational service within Development teams, helping them prevent and resolve production issues
- Allow continual service improvement through rapid incremental releases

Operations teams will usually operate an existing set of processes, often based upon the IT Infrastructure Library (ITIL) framework. The objectives above are deliberately couched in ITIL terminology, and should help to demonstrate that ITIL and DevOps are not at all in conflict – quite the reverse, in fact.

DevOps is heavily associated with a set of techniques collectively known as Continuous Delivery, which is a synthesis of concepts from Lean production, automated testing, Continuous Integration and Continuous Deployment. Although the two terms overlap substantially, DevOps tends to emphasise culture whereas Continuous Delivery focuses on specific techniques. They are mutually supportive:

- Continuous Delivery is most easily achieved once a DevOps culture and organisation is in place, otherwise it may not be politically or contractually viable to undertake deployment automation activities requiring facilitation by Operations teams.
- Adoption of some Continuous Delivery techniques – in particular automated non-functional testing, deployment automation, and Kanban systems – help to align Development and Operations teams' objectives. Using these techniques will assist with, but not guarantee, the establishment of a DevOps culture.

This clearly has the potential to be a Catch-22 situation. The good news is that if the Catch-22 can be overcome, then a virtuous circle can be brought into play where improvements in continuous delivery improve the relationship between Development and Operations, and vice-versa. Many organisations may find careful introduction of Continuous Delivery techniques a good way to help 'bootstrap' a DevOps culture.

DevOps is also often associated with the use of Cloud infrastructure. While not an absolute requirement for a successful DevOps culture, Cloud technology – public or private – can catalyse the required change and lower the upfront investment bar. It also codifies the interfaces between Development and Operations through formal, standardised service definitions that have been battle-tested and tuned by DevOps pioneers.

In turn, many of the benefits of moving to a Cloud infrastructure will not be fully realised unless a DevOps culture can be put in place. Without this, delivered applications will not be suitably architected and designed for scalable, reliable automated operation, and the touted flexibility of the Cloud for Development team will be reduced by politics and turf wars.

The relationships between DevOps, Continuous Delivery, standard Operations team processes and Cloud infrastructure are explored in more detail in the following sections.

DevOps and Continuous Delivery

Continuous Delivery can help meet many of the DevOps objectives previously described:

Deliver IT services that provide strategic value to the business

The concept of the deployment pipeline and associated Lean techniques are designed to ensure that strategic value is tested early by completing the full development lifecycle of a Minimum Viable Product from requirements through to benefits realisation.

Expedite project delivery of new services, reducing project costs

Getting a minimum viable product out to real users as soon as possible means that benefit can be developed quickly. Deployment automation improves delivery velocity and reduces costs.

Deliver services that satisfy both the customer's functional and non-functional requirements

Continuous Delivery emphasises automated testing of both functional and non-functional requirements with every release.

Reduce operational service risks associated with changes introduced by projects

Continuous Delivery emphasises that operations and infrastructure representatives need to be an integral part of the cross-functional delivery team from the start. Chunking delivery into smaller increments limits the build-up of risk associated with major 'change the world' releases.

Enterprise DevOps

ALIGNING DEVELOPMENT AND OPERATIONS

The Lean techniques on which continuous delivery is based emphasise proactive fail-safing - or 'poka-yoke' as it is referred to in the Toyota Production System, which was the genesis of the Lean movement. IT analogues including building automatic recovery from failures and techniques such as reconciliation that can detect and help limit the impact of problems that do occur.

The same Toyota Production System emphasises feedback from the coalface – originally via 'andon' cords which could be pulled by any worker to rapidly notify management of quality issues or even to halt the entire production line, picking up problems early and preventing expensive cascading failures.

Some release decisions have more to do with management politics than a careful assessment of the risks by those with the best technical knowledge, and Operations teams will be all too familiar with the effects of code that has been 'thrown over the wall'. It is essential to ensure accountability throughout the delivery team for the decision to release their code to production, and for being on the hook for helping to manage problems that do occur – whatever the time of day or night.

However, with the best will in the world, mistakes will occur. It is therefore critical to have quality control and warning systems to enable fast response to any issues. These might take the form of log consolidation and monitoring/dashboard systems at both technical and business process level. Reconciliations between different parts of a distributed system are also important crosschecks that the system is behaving as expected.

Finally, with Continuous Delivery it is important to have the capability to roll back changes automatically, quickly and cleanly if a problem is detected. This often needs to be designed into a system architecture from the start.

Manage configuration effectively and accurately

Manually deploying a new release into production can often take many hours and hundreds of error prone steps. Compliance requirements ensuring segregation of duties often mean that this process is commonly undertaken in production environments by generalist Operations teams. The result is that the configuration of development, test and production environments diverge in subtle and unexpected ways, leading to strange errors that can be very difficult and expensive to track to their root causes. These errors tend to manifest themselves as non-functional production issues because they occur less predictably and are less likely to be caught during smoke testing. The only viable way of preventing this is through deployment automation, which is at the heart of Continuous Delivery.

Allow continual service improvement through rapid incremental releases

This is almost the definition of Continuous Delivery, which adopts Lean continuous improvement principles at its core.

DevOps and Operations Team Processes

As described earlier, the objectives of ITIL and DevOps are well aligned. However, problems can occur when actually attempting to integrate established Operations teams and processes more closely with Development.

A DevOps culture is not about moving from rigorous production processes to an uncontrolled environment where developers can push to production at any time. Instead, it is about optimising a necessary and sufficient set of control processes so that they can be executed quickly and efficiently.

This requires high levels of automation, but a common problem in many enterprises is that essential Operations processes are often implemented in a very manual way. For example, ITIL has a heavy reliance on an up to date CMDB. A DevOps approach to meeting this Operations requirement would be to have deployment scripts automatically update the CMDB as environments are provisioned, adjusted and decommissioned.

However, if the CMDB is an Access database behind a firewall with no defined API, implementing this will be challenging. Similarly, if environment provisioning requires physical repatching of networks, it will be impossible to fully automate.

A combination of pragmatic incremental adoption of DevOps techniques, judicious application of tactical workarounds and careful process re-engineering and automation will address these problems over time.

> Some release decisions have more to do with management politics than a careful assessment of the risks by those with the best technical knowledge, and Operations teams will be all too familiar with the effects of code that has been 'thrown over the wall'.

Adopting a Cloud infrastructure can solve many of them in one fell swoop, as we will describe in the next section, and is already on the roadmap for many organisations.

However, two more insidious problems remain:

1. As with any control system, there is the potential for abuse of Operations team processes for political purposes. They are sometimes used not for the benefit of the enterprise, but as a tool to exert control, cement power bases and fossilise organisation boundaries. Because introducing DevOps will often require sweeping changes to the Operations function, generating losers as well as winners, politics can come to the fore. As a result, attempting to change these processes can be a formidable challenge.
2. Given the extent of outsourcing of Development and Operations functions, many processes may be 'baked into' commercial arrangements and therefore hard to change. For example:
    i. An outsourced development organisation may be have a long term contract to produce annual maintenance releases via a Waterfall process, and may not have any existing Operations expertise, neither of which is conducive to DevOps.
    ii. Providing real-time developer (or even server operations team) visibility into the production networking environment will be difficult if the network provider is not contractually obligated to provide a standard SNMP interface, and their profit margins depend on per-request revenue to provide the data in Excel form.

For these reasons, enterprises will often find it difficult to implement a full DevOps approach quickly. A clear longterm roadmap and focus on necessary organisational change management are required to make progress.

DevOps and the Cloud

The automation of infrastructure deployment at the heart of DevOps has been technically feasible for some time. However, it requires a rare blend of operations, infrastructure and development knowledge - historically a scarce resource - as well as up-front investment which could not necessarily be justified for all but the largest organisations. Cloud can help with this by providing some or all of the necessary engineering in pre-packaged service or product form.

When looking to deploy a major new application today, many enterprises will examine Cloud architectures alongside the traditional approach of buying/custom building and deploying on physical or virtual servers. The various Cloud architectures support DevOps in different ways by insourcing or outsourcing different parts of the delivery organisations and different parts of the technical stack as summarised in the table overleaf.

DevOps Resources

In terms of skillsets, introducing DevOps requires access to scarce skills. As increased DevOps adoption drives demand, these skills will naturally be at a premium until supply adjusts in the longer term. Using packaged solutions and services can help reduce the shortfall. Identifying and nurturing internal talent is essential and cross training will help. However, for many organisations it may be necessary to recruit from outside the organisation or use consultants.

Skill	Reason for scarcity
Developers who understand infrastructure and operations and automation focused infrastructure engineers and Operations staff.	The distinction between the roles of development and operations means that most developers do not have the depth of operational knowledge necessary to undertake complex infrastructure automation tasks, and few infrastructure and operations experts have the requisite software engineering skills.
Experienced automated non-functional testers.	Although functional testing automation skills are now relatively widespread, DevOps can only work if testers are able to automate testing for the factors that matter most to operations, i.e. performance, reliability, resilience and security. Few currently have this experience.
High-end technical management.	Any DevOps programme of enterprise scale will require first class, highly technically aware delivery managers and architects to plan and manage all of the necessary development, infrastructure, vendor management and organisational changes required.

Enterprise DevOps

ALIGNING DEVELOPMENT AND OPERATIONS

	SaaS	Public PaaS	Public IaaS	Private IaaS	On-premise PaaS	On-premise IaaS
Examples	Salesforce Sales Cloud, NetSuite, BigMachines	Azure, Amazon services including Elastic Beanstalk/RDS/ Dynamo, Force.com, Google App Engine	Amazon EC2, Rackspace, Azure, Joyent Public Cloud	Amazon VPC, Rackspace Hybrid Hosting, HP Virtual Private Cloud	VMware Cloud Foundry, Oracle Fusion Middleware, custom rolled stacks	VMware vCloud Director, OpenStack, Joyent SmartDataCenter, VCE
Descriptions	Internet hosted on-demand business applications	Internet hosted, pre-ntegrated end-to-end software stacks to manage deployment and operation of custom developed systems	Internet hosted, pre-integrated infrastructure software stacks to manage deployment and operation of standard Enterprise operating systems	Externally hosted on private infrastructure, pre-integrated infrastructure software stacks to manage deployment and operation of standard Enterprise operating systems.	On-premise hosted, pre-integrated end-to-end software stacks to manage deployment and operation of custom developed systems. Can be deployed on traditional datacentre hardware.	On-premise hosted on private networks, pre-integrated infrastructure software stacks to manage deployment and operation of standard Enterprise operating systems.
Infrastructure DevOps expertise	Outsourced	Outsourced	Outsourced	Outsourced	Either if custom-rolled, outsourced if using productised stack and appliance IaaS.	Either if custom-rolled, outsourced if using productised IaaS stack
Operations DevOps expertise	Outsourced	Outsourced	Outsourced	Outsourced	Either according to preference	Either according to preference
Development DevOps expertise	Outsourced	Outsourced	Either according to preference	Either according to preference	Either according to preference	Either according to preference
Infrastructure/Ops Management	Outsourced	Outsourced	Either according to preference	Either according to preference	Either according to preference	Either according to preference
Development	Outsourced	Either according to preference	Either according to preference	Either according to preference	Either according to preference	Either according to preference
Supplier management	Insourced	Insourced	Insourced	Insourced	Insourced	Insourced

It is important to have buy-in and oversight from the highest levels in the IT organisation, both to ensure the team can be protected from the political environment and to ensure an appropriate level of governance.

DevOps Implementation Models

The way in which DevOps is adopted will differ for each enterprise, and will have little to do with technology and everything to do with managing organisational change and acquiring the right skills.

We see five practical models for successfully implementing DevOps in the enterprise, which can be deployed individually or in combination.

1. Build out from the Operations team

This approach is an incremental development from the shared virtualisation facilities often already in place. Technically this will likely involve the introduction of an internal IaaS Cloud, with Cloud APIs and management tools initially only authorised to be used by Operations. For projects involving large amounts of infrastructure deployment, operations may combine infrastructure provisioning using Cloud APIs with automated application deployment scripts created by the Development team. This approach largely leaves in place the organisational distinction between Operations and Development, so is a relatively easy proposition from a political perspective.

However, it requires Operations staff willing and able to undertake complex automation. They must also take the time to understand the applications being deployed. This may be most easily achieved by handpicking individuals with a wide range of hands-on skills and high levels of infrastructure access, and seconding them into large projects. This is not easy - these people will be some of the most valuable and busy individuals. In highly specialised Operations functions with different application support, desktop support, server, storage and networking teams they may not exist at all.

If successful, the central operations team can be slowly restructured into a set of hardcore central IaaS infrastructure specialists, along with a services team effectively acting as a consultancy/QA function for projects. The key point is that this latter group must be genuinely service-orientated individuals, otherwise the old divisions will return.

In this approach, a traditional Operations function will be retained for a long time to maintain the bulk of legacy systems. As new systems are delivered, the transition can be gradual and managed through staff retraining and natural turnover.

2. Set up a DevOps skunkworks

If the organisation needs to see proof of DevOps working before it commits, then commissioning a dedicated team of skilled developers and operators to deliver a single DevOps system, including all associated infrastructure, is a good step. These could be seconded from the existing teams - in which case, they will need to be cherry picked - or recruited specifically for their DevOps skills.

This team must then be exempted from the need to conform to existing standards, which would likely be used as a lever to bring back the old divisions by those seeking to protect their turf and maintain the status quo. This will probably extend to providing the team their own racks in existing datacentres, along with the access to set them up and operate them, or letting them use an external Cloud.

As a result, it is important to have buy-in and oversight from the highest levels in the IT organisation, both to ensure the team can be protected from the political environment and to ensure an appropriate level of governance.

3. Build out the Development team

It may be less sensitive to allow development teams to pursue their DevOps ambitions in Dev/Test environments rather than with production systems. Operations can provide a Dev/Test IaaS Cloud to project teams, along with full freedom to create and destroy instances at will. As with all IaaS implementations, the enterprise will need to have sophisticated chargeback and capacity management systems in place, as well as good isolation between projects and very strong isolation from the production environments.

The project teams then become responsible for developing or hiring their own operational experts to bed down a DevOps culture on a project-by-project basis and for optimising their own environment usage and working practices. The good news is that projects are often good at this type of task.

Once the projects have automated their own deployments in Dev/Test environments, the scripts can then be handed over to a cadre of DevOps experts within the Operations function. Once fully productionised, these can be maintained on a joint basis.

This approach is again relatively easy from an organisational and political perspective, as it leaves the existing organisational boundaries completely in place. However it is not without its challenges:

- It does not address development visibility into production, and the resulting benefits of having a development team that understand how a production system really operates. As a result, while it may help with agility, it does not lower release risk to the same extent as other approaches.
- It is reliant on collaborative working between the operations function and the operations experts on project teams. If a 'them and us' culture develops, even the agility benefits will not be realised as teams quibble over the fine details of deployment methods.

4. Greenfield implementation

For those rare organisations that have the benefit of a green field site, they can save the pain of the transition by adopting Cloud technologies and automating testing from the outset.

In this model, the DevOps experience resides primarily with the PaaS and SaaS infrastructure providers. The rest of the Operations function is addressed directly by the Development and Application Support teams, which do not directly interact with the infrastructure at all: they simply roll a package and the team's release manager decides which test and production environments to deploy it into. All of the development, test and production environments are hosted on the outsourced Cloud and any infrastructure issues are dealt with via call escalation to the external provider. If an internal Cloud is used, a small operations function will still be required.

This approach will be particularly attractive to startups because it can dramatically cut time to market and provide immediate access to state-of-the-art infrastructure at low cost. However, as attractive as it sounds it has little relevance to most established enterprise IT functions due to the necessity to support legacy systems which will not immediately be compatible with PaaS.

Enterprise DevOps

ALIGNING DEVELOPMENT AND OPERATIONS

5. Force DevOps onto Operations

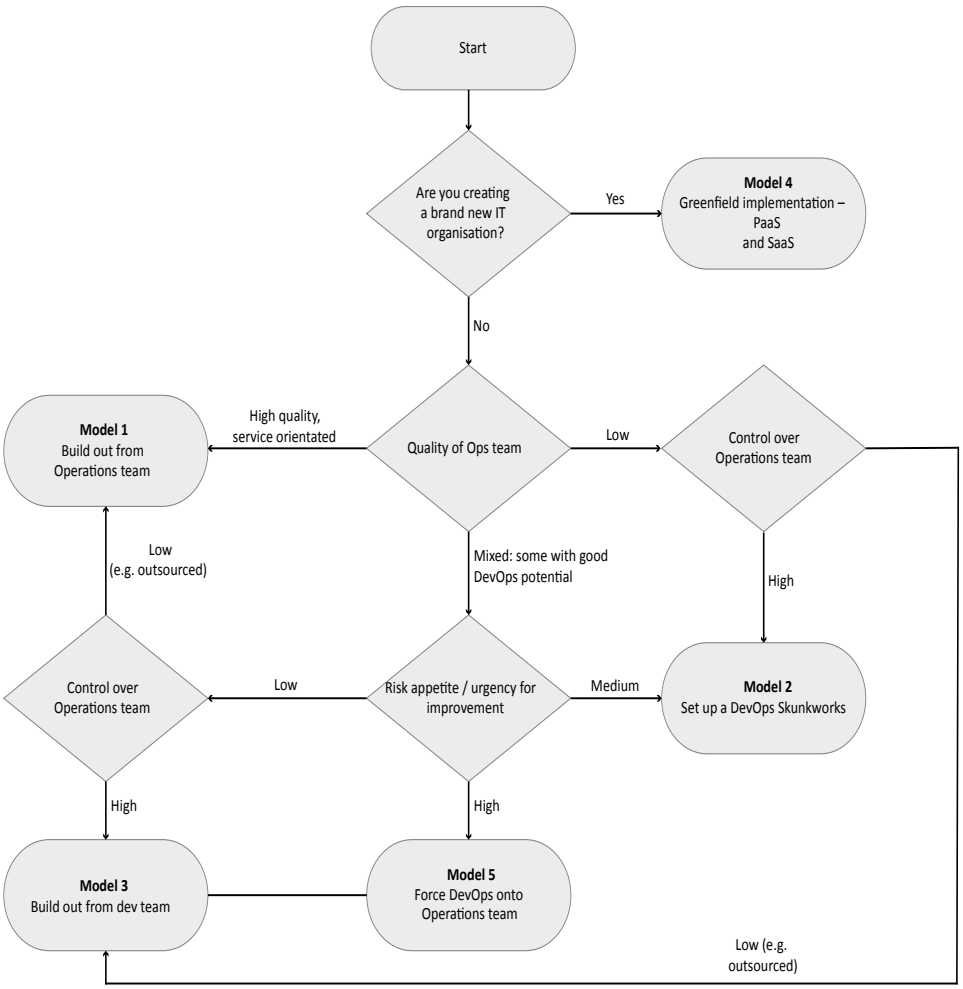
This is the riskiest strategy and the least likely to succeed: just adopt DevOps as your strategy, cherry pick members of the operations team who can act as advocates and can hold the course when the going gets tough, and hope that enough of the rest of the operations team come with you. It is critical to choose the right advocates from operations for this to be successful.

It is important to realise that many members of the existing operations team may not be able to adapt to the new ways of working. Combined with the fact that any central Operations function will be left considerably smaller in the post-DevOps world, this will generate formidable organisational and political resistance.

This approach should not be used unless you are totally convinced you have a critical mass of DevOps-minded individuals in place, as well as the ability to access very skilled external resources if required. This team will have to engage in a 'race against time' to automate operations for the legacy systems before attrition of the remaining team causes critical production support issues.

Choosing the Correct Model

The flowchart below summarises some of the factors that may affect which model you wish to adopt.



Conclusion

The full benefits of Agile will not be realised in the Enterprise without a focus on the integration of Development and Operations. BJSS Enterprise Agile has stressed this from the outset, but in many organisations it is not yet delivering to its full potential due to deep rooted issues in the relationships between the two teams

Delivery if IT services is analgous to a production line - speed of delivery and quality of the finished article are determined predominantly by the weakest link in the chain. Continuous Delivery's application of Lean manufacturing principles to optimise the delivery chain is a natural step, but needs to be underpinned by cultural change.

We are encouraged by the adoption of DevOps as a common label for the topic and by the increasingly wide interest in it. However, we feel that adoption is being hindered by ambiguity and imprecision in the way the term is used – often as a result of vendors stretching it to and beyond its limit in order to 'DevOpswash' existing offerings – and a dearth of quality analysis on how to implement it within enterprise environments.

We hope this White Paper goes some way towards remedying this by synthesising a range of sources into a coherent and succinct definition of DevOps and its relationship to the IT landscape, and that our implementation models provide a useful framework for thinking about how to overcome the many challenges.

Of course, our investment in writing this is not entirely altruistic. A rising tide lifts all boats, but it also makes it more obvious which ones are speedboats and which are rusting hulks. By encouraging adoption of DevOps we hope to make the benefits of BJSS Enterprise Agile, which we have placed at the core of our business, even more evident than they already are.

BJSS is well positioned to help enterprises address short term skills shortages until the wider industry catches up. We have been practicing and preaching DevOps principles for many years – well before the term itself came into existence.

We welcome feedback and questions via info@bjss.com and look forward to hearing from you.

Bibliography

John Allspaw: "10+ Deploys Per Day: Dev and Ops Cooperation at Flickr"
http://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr

John Allspaw: "Reply to 'Ops, DevOps and PaaS (NoOps) at Netflix"'
https://gist.github.com/2140086

Luiz André Barroso, Urs Hölzle: "The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines"
http://www.morganclaypool.com/doi/abs/10.2200/S00193ED1V01Y200905CAC006

BJSS (various): "BJSS Enterprise Agile: Best Practices in Enterprise Software Development"
http://www.bjss.com/enterprise-agile/form.php

Adrian Cockcroft: "Ops, DevOps and PaaS (NoOps) at Netflix"
http://perfcap.blogspot.co.uk/2012/03/ops-devops-and-noops-at-netflix.html

Cutter IT Journal: "DevOps: A Software Revolution in the making?"
http://www.cutter.com/promotions/itj1108/itj1108.pdf

Damon Edwards: "DevOps is not a technology problem. DevOps is a business problem."
http://dev2ops.dtosolutions.com/2010/11/devops-is-not-a-technology-problem-devops-is-a-business-problem/

Jeff Hodges: "Notes on Distributed Systems for Young Bloods"
http://www.somethingsimilar.com/2013/01/14/notes-on-distributed-systems-for-young-bloods/

Jez Humble and David Farley: "Continuous Delivery"
ISBN 978-0-321-60191-9

Mike Loukides: "What is DevOps?"
http://www.amazon.co.uk/What-is-DevOps-ebook/dp/B0084HJB56/ref=sr_1_1?ie=UTF8&qid=1358531733&sr=8-1

Pascal-Louis Perez: "Applied Lean Startup Ideas: Continuous Deployment at kaChing"
http://www.slideshare.net/pascallouis/applied-lean-startup-ideas-continuous-deployment-at-kaching

Eric Ries: "Lean Startup"
ISBN 978-0-670-92160-7

Chuck Rossi: "Release Engineering at Facebook"
http://devops.com/2012/11/08/release-engineering-at-facebook/

Mattias Skarin: "Introducing Kanban in Operations"
http://blog.crisp.se/mattiasskarin/files/slides/introducing_kanban_in_operations.pdf

About the Author

Martin Maisey
CTO, Consulting Division, BJSS

Martin has been a proponent of software release automation for over 13 years after taking over as technical lead in a development project that was failing in its system test phase due to environment issues. He took the decision to make deployment automation the priority of the 20-person technical team for one week, turning around the project. He went on to be an early adopter of continuous integration and was production systems architect for a mission-critical trading environment comprising over 50 integrated applications, including a $3m re-engineering of the infrastructure, high availability and disaster recovery architecture. In 2006 Martin built and led what would now be described as a DevOps team for a Software-as-a-Service project. He combined software and infrastructure engineering within a single unit to automate deployment of a fully configured test or production document management application within minutes from bare metal or bare virtual infrastructure. The system integrated to operational management systems including monitoring, configuration management, network provisioning, backup and problem management.
His most recent consultancy engagement was being brought in to assist a project delayed for six months due to major issues discovered at a late stage of operational acceptance testing. After three months of his involvement, the project delivered successfully into production.

LinkedIn: http://lnkd.in/gBKkPa

About BJSS

BJSS is an award-winning delivery-focused IT Consultancy.

With over 20 years' software delivery and IT advisory experience, we are renowned for technical excellence, cost-effective delivery and our proven BJSS Enterprise Agile approach.