InfoQueue
eMag Issue 14 - June 2014

Introducing DevOps to the Traditional Enterprise

Damon Edwards: DevOps is an Enterprise Concern

Damon argues DevOps is most needed in the enterprise world, and suggests starting with self-service provisioning interfaces, service oriented mentality, designing tool chains and meaningful metrics. All based on his own experience on the field as a DevOps consultant.

PAGE 4

Contents

A Letter from the Editor

Manuel Pais is InfoQ's DevOps Lead Editor and an enthusiast of Continuous Delivery and Agile practices
Manuel Pais tweets @manupaisable

DevOps, DevOps, DevOps. In 2014 DevOps hit the top of the charts in IT lingo. A recurring question that tags along is: can DevOps be applied in traditional enterprise settings?

Detractors say traditional enterprises' organizational and technological complexity, along with strong dependency on tool vendors, makes it hard to engrain and stick to the ideals of DevOps.

Supporters counter that organizational complexity leads to silos and lack of collaboration between dev, ops and other areas of work involved in the application lifecycle. Thus DevOps is needed to break those silos and instill a collaboration culture.

For this eMag we selected a set of articles that dig deeper into this debate and contextualize the benefits and challenges of DevOps adoption in a traditional enterprise.

In our interview with Damon Edwards we hear about his conviction that traditional enterprises are the ones that need DevOps the most as they face competition from upcoming smaller companies without the burden of legacy software and processes.

Jeff Sussna talks about the focus shift from long product development cycles to fast-paced services running 24x7 and how that revolutionizes the operational landscape along with the advent of cloud.

To quickly and efficiently deliver software and services, enterprises need frictionless handoffs and clear pipelines of work moving through the lifecycle. This means not only moving towards a DevOps mindset but also having an adequate software delivery strategy in place. Andrew Phillips shares some stories and pitfalls he found when implementing continuous delivery in enterprise settings.

Kamal Manglani & Gerald Bothello offer a hands on check list of hurdles one must overcome and commitments one must make in the journey towards a successful DevOps transformation. They know because they got it done at Walmart.

Introducing any kind of change in an organization is hard, and DevOps is no exception. John Clapham has been there and he offers a quick how to get started. In his article you can find many good resources on change management.

Finally, in the closing virtual panel several leading DevOps practitioners such as Gene Kim (author of "The Phoenix Project"), Jez Humble (author of "Continuous Delivery"), Patrick Debois (DevOps "founder"), John Allspaw (author of "Web Operations") and Mitchell Hashimoto (creator of Vagrant) come together and share their views on the challenges of adopting DevOps in the enterprise world.

Damon Edwards: DevOps is an Enterprise Concern

InfoQ: There has been some debate over the need for "big DevOps" for traditional enterprises. What's your take on that?

Damon: I find that a curious point of view. I mean, first of all look at what actually causes DevOps problems: if you get right down to it, it's silos: silo thinking, silo tooling, silo processes. This idea that any place there is a handoff, that is where the bottlenecks, that is where the misunderstandings, that is where the issues occur and who has more silos but enterprises?

So, in many ways, DevOps is really an enterprise concern. Large organizations, large enterprises – they have the history of success, the legacy of success: they have legacy people, legacy processes, legacy organizations, acquisitions they make and these silos build up very quickly and they are the ones that actually need the DevOps techniques more than the higher-flying sort of young web startups that are a single purpose organization that don't have to deal with that legacy history.

InfoQ: You are an advocate of Ops self-service platforms. What are the implications for large enterprises and how would you address the fear that a self service approach might lead to lack of necessary control?

Damon: If you think about the best way to remove the silos, the best way to get an end-to-end value stream working in an organization is to limit the number of hand-offs that have to take place. But in a large enterprise, you are just going to have different teams, different groups, you are going to have acquisitions, you are going to have various business lines feeding the common operations group, you are going to have geopolitical issues at places where people are placed in the world for cost factor and just based on where you can find them. So, these hand-off points are going to happen.

I think if you look at what causes these silos and these issues, it's any place that you have a kind of manual request queue. Think about what a terrible thing a ticket system is: you got some requests, you put it in there, somebody gets the request off there, they read the request and they have to interpret your question, then they have to ask you some questions back and you go back and forth and they send you something which is not quite right, you have to send it back. That whole time, all you are doing is putting extra chances for mistakes and you are putting extra cycle time and wait time into the process and these bottlenecks start to form.

Then on top that, you look at operations groups, they are hopelessly outnumbered. There is always going to be far more developers, far more testers, far more product managers and business folks that play there, all trying to feed into their kingdom, all trying to get into those operational capabilities. So, the idea for the self service thing is to say they need to turn those typical tasks, all those requests they normally would

fill–turn them into self service capabilities that the rest of the organization can pull as needed.

That's a massive gain in efficiency and it's a massive gain for the throughput over the organization because you've now allowed people to decouple, to solve their own problems and to move at their own pace and you get operations out of the task of day-to-day doing, digging through the muck and doing the activity and focused on higher level goals, how to actually improve the reliability, the stability and the scale of the organization.

Operations are being squeezed between moving faster and being more compliant. If you're using an automated system and you actually have the request going through this self-service system, you not only know who requested it, you have the automation that knows exactly what ran and you are able to actually bake security and compliance and governance into the platform.

InfoQ: Organizations today are shifting from delivering software products to delivering running services. What is the impact of this paradigm shift in operational terms?

Damon: The important thing to keep in mind is that if we are developing running services, then we need to make sure that from day one we are actually developing a running service.

I believe software becomes a service when: number one – it is running; and number two – it is fully managed.

So, we have the automation in place to deploy it and configure it in the way that it is repeatable and anybody can do it. And we have all the standard operative procedures automated, how to start it and how to stop it, restart it, reset it, reconfigure, check if it is healthy, that type of thing. And we have all the monitoring and metrics and visibility in place to know the health of that service and make it manageable.

So, if all those things in totality equal the product, they are equally important as the code itself. Do those things early in the process, make them part of the developer's deliverables. Have them use the same tools, the same kind of tests, the same kind of automation that would be used in a production environment in their development environment.

So we have to think about a running service from step one, we cannot wait for that to be an operational concern that happens down the line. The high performing companies that do that - the throughput and the speed and the quality which they move – it's a stark difference between that and what you see in these more traditional organizations where the software developers write software, that is their job, and then it's somebody else's job to test it, somebody else's job to automate it. They are the ones that are having the quality and throughput problems that you hear about so much.

InfoQ: Why is the design and thinking about tool chains important for DevOps?

Damon: We teach organizations to think about a tool chain approach, so that there is a design pattern that everybody can agree on standardize on. Then the individual points on that tool chain and the tools that you use become a very simple to select depending on the tool meeting the design requirements.

We look at very successful organizations - they take this kind of loosely coupled tool chain approach. They say it's more like a Unix style tool chain where you basically take tools that are good at individual things, that are the simplest, easiest way to accomplish something and worry about integrating those tools with each other.

Asking "What is the tool chain we need to tie our application deployment to our infrastructure provisioning? How can we manage those things in a consistent logical way?" allows you to have conversations, or arguments if you will, about the design and the rest becomes a simple selection process to plug things in.

If you are able to have a coherent and consistent tool chain design throughout the organization, then it becomes easy when you talk from group to group to say: "For this particular function we use tool X and we have the same type of integration points with this tool Y that's the same as you use".

So, it is easy to understand how those parts are swappable, it is easier to understand what the role

of the tools is and the organization can worry about getting the work done versus having these religious battles of "is it tool X or is it tool Y?". That really does not serve anybody; it doesn't really help the company.

InfoQ: Traditional organizations often try to standardize on the same suite of tools and everyone has to fit their way of working into the tool. Do you think that's a good approach?

Damon: I think, first of all, what is the point here? Is it the point to standardize for efficiency or to cut down on the number of vendors we have to deal with or is it the point to improve the time of market and quality or your product and the effectiveness of your organization? That is what actually matters to the business.

So, if limiting choices helps you do those things, then that's good, but in many cases those choices are constraining the organization on a tool basis first, that does not help because you are actually constraining the people doing their work and you are constraining the process that they are trying to get things done.

If you solve the people problems, if you solve the process problems, then the tooling problems should be straight forward engineering decisions. If you have the right kind of people and process ideas in place and you have the right tool chain design in place, then the organization will make the right selection when it comes to the tools and people will be effective and move forward.

InfoQ: In your talks you say the first step to improve the workflow in an organization is to visualize the system. How do you achieve that in large organizations with multiple, often non-collocated, departments?

Damon: First of all, I think there is no substitute for human-to-human connection at this point. I think you need to get people on the same page as to what is actually the process, the flow of work through the organization. People see a couple of things in large organizations. They see an organization chart, but that is just a collection of people and dotted lines and it does not actually tell you how work flows through the organization. Secondly, they see the physical systems.

They have a lot of technical metrics about the machines they are now running, they have some code metrics and information about their code and source code and lines of code and the various tests and the build telemetry they've got.

Then they have a bunch of financial data: we made so much money or we lost so much money. That is it. What is actually missing there is the visibility into the activity and the flow of work through the organization.

First and foremost, you need to get people aligned towards what is the flow of work in their organization, what is the end-to-end process, what is the value stream. Looking at artifact flow: how do the artifacts flow through the organization? And looking at information flow: what are our communication lines? What documents need to be passed? What is the signaling that has to take place?

The second thing is to raise awareness of the actual flow of activity through the organization. On the program/project management front, things like Kanban – they are great for watching the flow of work, understanding where things are and how things are moving through the life cycle, where the bottlenecks are, which resources are overcommitted versus under-committed or they are at capacity.

In low performing organizations nobody really has an idea as to how work flows through the organization. The high performing organizations have their deployment pipelines, they have the value stream maps, they have a lot of information about how work is flowing through the organization and, not surprisingly, people respond to it and self organize to improve that flow.

InfoQ: Meaningful metrics should provide visibility on how the overall system is doing. How can you get meaningful metrics from operational, development and business point of views simultaneously?

Damon: I guess the point of any metric is what behavior are we trying to encourage? For getting an organization to improve how they are working, I think you need to limit the number of metrics that

you are asking people to look at to a few meaningful things that actually can move the organization.

There is always some higher level business metric that needs to be accounted for. The Amazon guys talk about their order rate. It should be on everyone's desktop. So everyone is going to have a business metric that matters, that they need to talk about, but if you want to look at improving the flow of work through the organization, improving throughput and improving quality, solving your DevOps problems, how do you know you are getting better?

I noticed that there are these four metrics that really help organizations improve. The first one is obvious – cycle time. If you talk about the most minimal change from a business person's head to a running feature in a customer facing environment making me money, how long would that minimal change take to get through their entire process?

The second is meantime to detect. How long does it take you to actually detect if a problem is happening? That gives us a sense for what kind of handle do we have on the system, how well do we understand it, what is our monitoring, how well are things decoupled and isolated so that when a problem happens we can pinpoint exactly why, where it is and have a traceability back to understand what caused that.

The third is meantime to repair. So, if we know what is wrong, how quickly can we recover from that? That speaks to how automated, how cleanly, how configurable and manageable the system actually is – our ability to actually repair, well first of all to restore.

The last one is a bit more esoteric which is quality at the source. We know that the further a bug gets down the life cycle, the more expensive it gets to fix, the more knock-on problems and issues you are going to have. So, how quickly can we catch those errors – that speaks to our ability to test, it speaks to the kind of fast feedback we have and there are two ways to look at that.

One is the notion of "scrap", meaning how often does something go to the next step in the process that is not quite right. It does not have to be an outage. It could be like: "Hey, I provisioned a QA environment for you." You log in and you say "Oh, this is all wrong. I have to change this, change that" or you send it back. That's scrap. Think in the manufacturing sense: I made a bumper, I made ten bumpers, I gave them to you, they do not fit on the car. That is scrap, we have to throw them away or go back and rework them. So, that is one.

The second part of that is that when scrap does occur or problems do occur – were they caught? How far does a problem get down the life cycle, is it one, two, three degrees away from the source? The further it gets away from the source, the larger problem we have, so we want to keep those numbers close to 1 or 0, to say that problems are caught through fast-feedback, good testing at the source, before they move on the lifecycle.

So those four things: cycle time, meantime to detect, meantime to repair and quality at the source. If you track those things, then you are going to be able to know whether you are improving as an organization.

InfoQ: DevOps has been under the spotlight lately. As a DevOps consultant for multiple organizations, how do you assess the actual DevOps adoption in the field?

Damon: It is interesting. I think it is really just beginning.

I think there are three groups out there. There are the (web space) folks that have always been doing this; they have always sensed there is this bigger problem.

Then I think the next group is folks that saw that and thought: "Hey, how can we translate that to our kind of large organization?"

And the third group I think is what we are seeing now, which is the enterprise really realizing that time to market and quality is a killer. If you do not fix those problems, they will eventually put you out of business in this new hyper-competitive world that we are in.

The analysts are now explaining to the C-level executives that this business concern that you do not know how to fix, the time to market and quality problem that we seem to be suffering from – this area of practice called DevOps actually has some meaningful solutions in this space. And they're actually able to point to these higher performers to

say: "they know how to move the organization to do great things, we have to inject those ideas as well".

InfoQ: You recommend devs to walk in ops shoes, to understand what goes on in operations teams. How can this be achieved in organizations with a lot of silos and often externalized operations?

Damon: Traditionally software developers had a certain culture in the way they've worked which comes from the software package world. We write software, we test our software and then we ship our software. It is somebody else's problem, it is in a boat, it is gone. And that is operations, customer service, some other person on the other side that takes care of it, right?

And I think that in this new service oriented world, it is not the case. First of all, you are never done, the service is never done until it is turned off so this idea of shipping does not really make a lot of sense anymore.

So, first of all on the development side we need to get in that operational mind set to realize that this whole company is an operations company. There is no Dev and Ops, our whole job as a company is to operate software. So we are all operations in some way. So the thing is approaching Ops with that mindset and understanding that, especially in enterprises, there is this dual pressure on Ops and all we see is one side, which is the pressure to change faster. Well, what is also the biggest thing in IT in the news right now, it's compliance, it's security, it's risk. Plus you have regulatory compliance on top of that as well.

So, we have to put ourselves, from the development perspective, in operations shoes and say "How can we actually meet those requirements? How can we actually help them be more secure, be more compliant?"

So, I believe that is the responsibility of development, not just to say: "Oh, you know, Ops is terrible, Ops always says no, Ops never lets me do anything. Give me root and give me access and I will go and fix everything". It is not productive and it is not realistic. There is a real concern behind the reason why they do things, so I say that if you want to start from a development organization and make these changes a reality, you have to get your mind straight in terms of approaching it as a fellow Ops traveler who just has a different skill set and different day-to-day focus but we are in this together and we need to build a better future.

InfoQ: Do you think it can make sense to have a DevOps team bridging the gap between Dev and Ops in cases where there is actually physical distance and sometimes externalized services?

Damon: First, you cannot separate the people management from the technology management. I think that is a major mistake and you see high performing organizations – they do not do that. They get everybody thinking about culture, everybody thinking about the organizational and human dynamics that go into their work. You see teams that look at different far-flung disciplines and they are bringing them into their organizations to say: "We need to learn how to work together better as a team". They see themselves as a team in the organization, is the thing that succeeds.

That being said, I think that it is dangerous to say we have a DevOps team. If you say "That is the DevOps team. They are going to solve these problems." is like saying that we have silos so therefore we are going to create a new silo to solve our silos. It does not really add up.

That being said, I have seen DevOps teams be successful is when that team is really an architecture, tooling advisory, kind of coaching team that goes throughout the organization and looks for bottlenecks and looks for bad handoffs, looks for issues and helps to apply some muscle and apply some dedicated thinking to help those different teams to work better. That can be successful but the problem too quickly becomes like: "Oh, they now start building their own tools, they start becoming this other silo or kind of heavy handed saying you must do it this way". It goes wrong very quickly so I think in that way is best to avoid saying "This is the DevOps team".

To go back to the physical distance, you need to prove that you can build confidence and a lot of times just focusing on getting control over the Dev and QA cycle to say: "We are going to show you how quickly we can turn through these dev and test cycles and we are going to manage all these new practices and all these new tools just to QA" – basically like a continuous delivery type of scenario. Then use that

as a building block to show the operations side: "Hey, why don't we extend this to production?" Still deploy things the old fashioned way. Do it the way you think it is best, but please run my tools as well, next to it, and then I will show you all these automated tests. I will show you these validations/verifications. Once you get confidence to see how that works, then you can say: "hey, why don't you also push that button and deploy it as well, too"

InfoQ: How can someone in an organization trying to move towards a DevOps culture know if they are moving forward?

Damon: Number one: we talked about those metrics before – cycle time, meantime to detect, meantime to repair, quality at the source. Those are like service delivery ideas. How good are we at delivering the service that we are working on?

I think focusing on that is good because it focuses people on the goal. Looking at it from a more qualitative perspective, I think it is important, especially for managers who kind of want to get a sense for how are things actually going, to sometimes just ask somebody: "What is wrong?" and actually mean it, actually want them to tell you what is going wrong or right in an organization.

ABOUT THE INTERVIEWEE

Damon Edwards is the co-founder and managing partner of the DTO Solutions consulting group. Damon is also a frequent contributor to the Web Operations focused dev2ops.org blog, the co-host of the DevOps Cafe podcast series, and a co-author of the DevOps Cookbook from IT Revolution Press.

WATCH THIS INTERVIEW ONLINE ON InfoQ

Cloud and DevOps: A Marriage Made in Heaven

The Digital Innovation Economy

What is the relationship between cloud computing and DevOps? Is DevOps really just "IT for the cloud"? Can you only do DevOps in the cloud? Can you only do cloud using DevOps? The answer to all three questions is "No." Cloud and DevOps are independent but mutually reinforcing strategies for delivering business value through IT.

To really understand the relationship between cloud and DevOps, it's helpful to take a step back and consider the larger context in which both are happening. Cloud and DevOps have evolved in response to three fundamental societal transformations. First, we are in the midst of a transition from a product economy to a service economy. People are placing less emphasis on things and more emphasis on experiences. While companies still produce products, they wrap them inside services. BMW includes routine maintenance in the price of a new car. Cadillac integrates the OnStar service into its vehicles. Much of the power of the iPhone comes from its integration with iCloud and iTunes.

The transition from products to services is impacting software delivery as well. Previously, development companies built software products, and delivered them to customers who took responsibility for operations. With the advent of cloud, the majority of companies that build software also operate it on their customers' behalf.

Software as service is happening at all layers of the IT stack. At the bottom, infrastructure-as-a-service delivers on-demand virtual machines, networks, and storage. Platform-as-a-service delivers on-demand databases, caches, workflow engines, and application containers. Software-as-a-service delivers on-demand business functionality. At every level, providers allow customers to consume services based on demand, pay for them based on consumption, and offload responsibility for their management to the provider.

Second, the 21st-century business environment is forcing companies to shift their focus from stability and efficiency to agility and innovation. The pace of disruption is accelerating. Kodak lasted a hundred years before having to face the erosion of its place in the market. Microsoft, on the other hand, felt the ground shift under its feet after only thirty. Apple has gone from the most valuable company in the world to a question mark in just a couple of years.

In order to present an adaptable face to the market, companies need to change their approach to work. They need to shorten work cycles, increase delivery frequency, and adopt an attitude of continual experimentation. Social media is shifting power from producers to consumers. Marketing is changing from a process of driving behavior to one of responding to it. From the corporation as a whole down to the individual employee, companies need to empower creative responsiveness, and minimize any waste that impedes the ability to act on it.

Third, the digital dimension is completely infusing the physical dimension. Is your car a vehicle made of metal and plastic or is it a Pandora music-service client device? Is your office building HVAC system a marvel of fluid dynamics or a marvel of big data? Is your local library a place to find books on the shelf or a place to look them up online? The answer, of course, is "Yes."

Digital infusion dramatically raises the stakes for IT. We're reaching the point where daily activities are becoming impossible without digital technology. Companies depend on IT for their very existence. IT can't afford to fail at providing a compelling platform for the adaptive business.

Enabling Agility

What do these transformations have to do with cloud or DevOps? Cloud is a direct response to the need for agility. Initially, people saw cloud primarily as a way to save money and transfer capital expenditures to operating expenses. They've since come to realize that its real value lies in reducing waste that impedes speed and defocuses effort. Very few companies would identify data-center operations as part of their core-value proposition. Cloud services let IT departments shift their focus away from commodity work, such as provisioning hardware or patching operating systems, and spend it instead on adding business-specific value.

The transformation from product to service economy, along with digital infusion, means that companies need to become software service providers as well as consumers. I've reached the point where 99% of my interaction with my bank takes place via their website or mobile app. I judge their brand by the quality of our digital interactions. I judge those interactions across the dimensions of functionality, operability, and deliverability. I expect seamless quality across all three dimensions.

Cloud enables greater business agility by making IT infrastructure more pliable. It lets companies conduct digital service relationships with their customers. Cloud is only part of the answer, though, to the question of how IT enables adaptive businesses. Whether an IT organization runs a company's applications on data-center hardware or on a private or public cloud, it still needs to align itself with the business's needs, rather than forcing the business to align itself with IT's. Silo-based organizations and manual processes still create waste that impedes the ability to deliver continuous change and conduct continuous experiments. Onerous, time-consuming, arms-length change management procedures still generate resentment and frustration, and lead users and developers alike to seek ways to get around IT altogether.

IT-ops organizations often get tagged with the unfortunate moniker "the Department of No". Frustrated businesses used to tag development with this same moniker. The agile development movement has made great strides towards creating mutually trusting relationships between business and development. Agile comes in many flavors and has its own imperfections. At its root, though, agile is about tuning development to be receptive rather than resistant to change.

The Inseparability of Functionality and Operability

From a DevOps perspective, the most important implication of software-as-service is the way in which it dissolves the separation between function and operation. Users experience them as seamless aspects of a unified whole. At the same time as they expect high levels of functional and operational quality, users also expect service providers to deliver continuous change on top of that high-quality platform.

These expectations necessitate a fundamentally different approach to delivering software. Separating development from operations clashes with the outside-in view of inseparability. "Function + operations" maps more naturally to "development + operations". DevOps is exactly that. DevOps represents an effort to accomplish the same mutually trusting relationship for software-as-service as agile has done for software-as-product. Agile has taught development how to move at the same speed and with the same flexibility as business; DevOps tries to teach operations to move at the same speed and with the same flexibility as development. Success in the 21st-century requires radical alignment of goals, viewpoints, language, and cadence from marketing all the way through to operations.

Cloud and DevOps Aren't Just for Web Apps

What about IT organizations that work in regulated industries? Can they not use cloud? What about IT organizations that primarily operate commercial software instead of doing their own custom

development? Can they not adopt cloud or DevOps? The answer to both questions is "Yes, they can."

IT organizations need pliable infrastructure all the way from development to production. Centralized, shared development and test environments generate tremendous waste through polluted test data and contention for resources. IT organizations needn't wait for the ability to use cloud in production, whether public or private. They can use tools like Vagrant and Docker to improve productivity on top of desktops and shared test infrastructure.

Organizations that manage commercial software still need to coordinate function and operations. They also need to reliably, frequently deliver change, even if that change consists of business-rules configurations. Production support needs to understand the totality of change, from business rules at the top down to infrastructure at the bottom. These kinds of organizations can benefit from cross-functional collaboration, comprehensive version control, and automation just like any other.

The truth is, though, that a digitally infused service economy may make pure commercial application-support IT a thing of the past. As IT becomes more essential to business value, more companies will need to invest in some amount of custom development, if only at the integration and API level. Inseparable development and operations practices are universally relevant.

Cloud computing, agile development, and DevOps are interlocking parts of a strategy for transforming IT into a business-adaptability enabler. If cloud is an instrument, then DevOps is the musician that plays it. Together, they help IT shift its emphasis from asking questions like "How long can we go without an outage?" to "How often can we deliver new functionality?" or "How quickly can we deploy a new service?"

Adopting Cloud and DevOps

If you are a "legacy" IT organization that is struggling to adapt to new business demands, how do you take up cloud or DevOps? Do you need to shatter your org chart, or make a massive investment in deploying a private cloud? The principle of continuous improvement is key to agile, cloud, and DevOps. It should guide your approach to adopting them as well. Continuous improvement speaks about

"starting where you are". The truth is that there's nowhere else to start.

Adaptive business is about always asking questions of yourself:

- What's changed since we last looked?
- How can we get better?
- What can we do differently?
- What haven't we thought of?

Adaptive IT is about asking the same kinds of questions. Rather than reacting to new ideas or methodologies with "We can't because…", instead ask yourself:

- Why can't we?
- How can we?
- What's the first step towards getting from here to there?
- What can we stop doing?

These questions may lead you to do things like experimenting with a cloud platform for a single test environment, or inviting InfoSec staff to standups for a single project. Learning what works, and how it works for your organization, shows you how to propagate it more widely.

No Time to Waste

Just like the business as a whole, IT needs to engage in continuous experimentation. Public clouds like AWS, along with shadow IT, are pulling businesses away from internal IT departments. The time for fighting to retain control is past. Its urgently need to change from being the Department of No to being the Department of Look Over Here. Cloud and DevOps are two enabling practices that can help IT address the larger transformative shifts – the service economy, continuous disruption, and digital infusion - that are driving business in the 21stcentury.

ABOUT THE AUTHOR

Jeff Sussna is founder and principal of Ingineering.IT, a Minneapolis consulting firm specializing in continuous delivery, cloud computing, and design thinking. Jeff has more than 20 years of IT experience, and has led high-performance teams across the Dev/QA/Ops spectrum. He is a highly sought-after speaker on IT innovation topics, and was recognized as a Top 50 Must-Read IT Blogger for 2012 and 2013 by BizTech Magazine. His interests focus on the intersection of development, operations, design, and business.

READ THIS ARTICLE ONLINE ON InfoQ

Preparing for Continuous Delivery in the Enterprise

by Andrew Phillips

Introduction

Continuous delivery as a software-delivery strategy is attracting increasing attention and recognition within both IT and the business organization.

The ability to rapidly and repeatedly bring service improvements to market that continuous delivery provides aligns naturally with business initiatives to accelerate time-to-market in today's competitive economic environment, while maintaining quality. Frequent, incremental changes also help meet the expectations of today's "always on" consumer, whose work and leisure experience of IT services is increasingly based on simple, one-click installation of applications that update automatically.

Virtualization, private cloud, and DevOps initiatives are ongoing in many organizations, providing the technical foundations for continuous delivery. Combined with competitive pressure created by the growing, successful adoption by industry leaders, it is no surprise that surveys show that continuous-delivery implementation is one of the current key initiatives for many enterprises.

Where to Start?

As a concept, continuous delivery has been around for a while, and has indeed been practiced by the front-runners in this field for many years. As a result, plenty of books and other references describe the principles and practices of continuous delivery and sketch out in detail what the goal state can look like.

These materials can provide an important theoretical grounding and can help you develop your vision of continuous delivery. It is in getting started to realize that vision, especially in the context of the existing development and release environment in a large enterprise, that additional guidance is needed.

Our experience helping clients introduce continuous delivery and related automation, and partnering with other leading continuous-delivery experts such as ThoughtWorks, can help. We will describe important steps to preparing a structured implementation of continuous delivery that takes you from where you are today to the realization of your vision in well-defined, measurable phases.

Before You Implement: Identifying Potential Challenges

As in any practice-oriented implementation, a clear picture of the current situation is important. Awareness of your baseline and resulting challenges to the implementation are a prerequisite to successfully implement continuous delivery.

Based on our experience helping enterprises both large and small introduce continuous-delivery automation, we can describe a number of factors that can impact the implementation of continuous delivery.

Not all of these factors will be relevant to your scenario. For those that you do recognize, an action item to identify the scope of the challenge, its

potential impact on your delivery goals, and possible mitigation steps should be part of your project plan.

Enterprise Challenges

1a. Large, monolithic applications

A key aspect of continuous delivery is making small, incremental changes to your applications and services. These are simpler to test and, since only a small part of the application changes each time, make pinpointing and remedying the source of errors and other problems such as slowdowns much easier.

Since each change has a much smaller impact on your target environments than a typical "big bang" release, it can usually be pushed through your pipeline more quickly, too. This leads to faster pipeline runs, shorter dev-to-prod cycles, and, thus, steadier feature throughput.

Large, tightly-coupled applications in which many components need to be compiled, tested, and deployed together are hard to update incrementally, leading to long development, test, and deploy cycles. Quality control and especially root-cause analysis are harder, too, since many changes are being implemented at the same time.

The large number of changes and components that need to be modified mean that, typically, each release procedure needs to differ slightly from the previous ones. This makes it hard to create a standardized delivery pipeline and to benefit from the resulting increase in reliability.

As with any iterative process, improvement through fast feedback is a key part of a successful continuous-delivery setup. Drawing effective conclusions from one enormous chunk of feedback after a monolithic release is hard as there are so many changes and factors unique to this release to consider.

Smaller, faster, more standardized pipeline runs greatly simplify the feedback and improvement cycle.

1b. The story of the big app

As part of moving to a more agile development methodology with shorter iterations, a large insurance organization started building a delivery pipeline for a claims-processing platform. The platform consisted of a single application based on an internally developed framework and took over an hour for a full compile and unit-test run.

Initially, environment-specific endpoints in the source code required the application to be built separately for each environment, so that a build-functional test-integration test cycle required almost four hours just to build the deliverables!

As a first step, these environment-dependent values were externalized, allowing (in accordance with continuous-delivery principles) just one deliverable to be built. Weighing in at more than 1 GB, this artifact still took more than 20 minutes simply to be copied to the next pipeline stage, with a further 30 minutes for the target middleware to deploy and process it.

As a result, developers ended up committing changes more quickly than the pipeline could handle. In order to deal with this, the pipeline was throttled to trigger only once per day. This alleviated the bottleneck but meant that failures in the pipeline could no longer be related to individual changes, making it harder to fix errors quickly.

In order to address this challenge, developers initiated a work stream to incrementally break out components of the application into separate modules that could be built and deployed independently, allowing for faster feedback cycles with smaller change sets.

2. Low levels of automation

If a review of the various activities that your environment currently requires to transition a new application version from development to production identifies many manual steps, you may need to consider ways of increasing the level of automation in your pipeline.

It's not that we should be automating for the sake of automation: manual activities aren't banned from a continuous-delivery pipeline on principle. Experience simply shows that humans tend be both slower and less accurate at the kind of repetitive tasks that make up the bulk of a delivery pipeline.

A high percentage of manual steps will thus likely prevent you from being able to scale your continuous-delivery implementation to the desired number of pipeline runs. In order to meet your throughput and consistency goals, you are usually required to either automate many currently manual steps in your delivery process or remove certain

steps from the process altogether if suitable alternatives are available.

It is important to treat this automation effort as seriously as any other development effort, applying appropriate design, coding, and testing practices in order to avoid ending up with a ball of mud that's impossible to maintain. The infrastructure-as-code movement has made significant steps in this area, for instance promoting test-driven development of provisioning and deployment automation and providing supporting tooling.

3a. Contended environments

If your organization currently works with a limited pool of shared test environments, there is a risk that you will quickly run into a bottleneck while implementing continuous delivery.

Firstly, the ability to block or reserve an environment becomes necessary if your delivery pipelines trigger on code changes: two pipelines running side by side need to be prevented from attempting to deploy and test in the same environment. You also need to take measures to prevent one pipeline from blocking an environment for too long or to keep one pipeline from always just beating the other to the required environment, leading to resource starvation for the lagging project.

An interesting data point in the survey mentioned in the introduction is that one of the leading causes of deployment failures is a misconfigured or broken environment that has been unexpectedly modified by previous teams or test runs. Even if your environment pool is sufficiently large to avoid the starvation problem, regular pipeline failures due to misconfigured environments will also limit your ability to reliably deliver new features.

If you plan to run delivery pipelines at scale, you need a dynamic pool of available, clean target environments. Private, public, or hybrid cloud platforms, coupled with provisioning and configuration-management tools, allow you to grow and shrink this pool automatically and on demand.

3b. The story of the limited environments

After a significant push to develop automated tests, a retailer implemented a publishing pipeline for a large customer-facing website. Next to accelerated releases of new content, another anticipated benefit was improved utilization of the three test environments that, due to a complex integration of the CMS, webservers, and a number of external endpoints, had been complicated and expensive to set up. In order to prevent environment conflicts, a round-robin system had been implemented, with subsequent pipeline runs blocking until the next test environment became available.

With content changes arriving frequently, it quickly transpired that one suite of tests that verified long-running buying sessions caused the pipelines to back up and eventually overload the orchestrator. First, the retailer added a hard timeout that would allow the pipeline to resume. Subsequently, tests started behaving erratically, failing in some runs then succeeding immediately afterwards, without obvious cause.

Investigation revealed that the hard timeout was disrupting the database cleanup procedure, leading to corrupted environments. To prevent this, the long-running test suite was simply disabled, lowering test coverage. After a high-profile glitch, the team bit the bullet and set up automated provisioning for their test environment, including development of mocked-up versions for external endpoints.

Automated test-environment setup has since been a requirement for all test-automation projects at the organization.

4. Release-management requirements

The canonical delivery-pipeline diagram, with its stages and feedback all the way through to production, looks temptingly straightforward. In practice, as soon as we approach QA or production in most enterprise environments, an increasing number of release-management requirements must be met: creation of a change ticket; placing the change on the agenda of the next change-board meeting; receiving change-board approval; confirming deployment windows; etc.

How to integrate such requirements into our delivery pipelines is an important question that continuous-delivery implementations in enterprises need to address. One option is to simply cap all delivery pipelines at the test stage, i.e. before we run into any release-management conditions. The goal is typically to take continuous delivery further than just test environments, though.

Can the various release-management steps be integrated into the pipeline, e.g. by manually and, eventually, automatically creating and scheduling a change ticket or by automatically setting a start time on the pipeline's deployment phase from the change-management system?

Is it possibile to revisit the need for certain change-management conditions in the first place? Most change-management practices come from providing assurance that only changes of an approved level of quality and stability make it to production – precisely the level of quality and stability that prior stages of a delivery pipeline are intended to verify.

Experience from well-known examples of organizations proficient in continuous delivery, such as Netflix, Etsy, and others, indicates that quality, traceability, and reliability of releases can be achieved using pipelines, without the need for heavyweight change-management processes.

5. Scaling up jobs

A large organization with a diverse service portfolio has many pipelines to manage as it scales its continuous-delivery implementation. Your service portfolio likely spans different technology platforms, different departments, different internal and external customers, different development and support teams, etc.

If each application defines its own custom pipeline, how will that affect management and measurement of your continuous-delivery implementation as a whole? If every pipeline ends at a different stage in the delivery process, how can you compare metrics such as cycle time, throughput, or percentage of successful executions?

A large set of pipelines is easier to manage if each one is based on a standard template. Templates can be as simple as a shared wiki page but common pipeline orchestrators such as Jenkins (Templates Plugin), Go (Pipeline Templates) and TFS (build process templates) also support them. Standardized pipelines also allow for more meaningful comparative reporting as well as allowing lessons learned in one pipeline to be applied to many others – improvement through feedback being a key component of continuous delivery.

How many templates you should start with depends on the variation across your service portfolio; one per technology stack is often a useful starting point. Over time, you will hopefully eventually be able to consolidate towards just a handful of pipeline types.

6a. Job ownership and security

When everything is running smoothly, it is easy to forget that automated delivery pipelines are processes that span many parts of the IT organization, from development through testing to production.

When pipeline stages fail, though, it is essential that clear responsibilities are in place to fix things and get the delivery stream running again. Every pipeline stage should have an owner, champion, or responsible person/team who is tasked not only with fixing problems but also contributing to feedback-driven improvement of the pipeline as a whole.

Since visibility into the state of the entire pipeline is important for all stakeholders, not just the owners of the individual stages, it is important that any considered orchestration tool offers a suitable security model. For example, developers will probably need to examine the results of a functional test to help identify the cause of test failures. They should not be able to disable or modify the configuration of the functional-testing step, however.

6b. The story of the orphaned pipeline

Planning a greenfield project to develop a mobile application and corresponding backend to allow customers to customize their vehicles, a car manufacturer decided to start with continuous delivery from the outset. It hired a couple of experienced build-and-release engineers as consultants to set up the pipelines. The project successfully went live and the consultants moved on, job well done.

For a while, everything went smoothly. Then, one of the mobile platform branches suddenly failed. Cue angry calls and emails from the business owners, and a quick fix: the failing pipeline stage was simply cut out and bypassed. The (only partially tested) application appeared in the store again, and the missing stage was quickly forgotten.

Over time, similar emergencies resulted in disabling or reducing in scope more and more segments of the pipeline. Small bugs appeared in the application with increasing frequency, but without an owner and with

the original creators gone, restoring the pipeline to its original state was on nobody's radar.

A new member of the development team, coming from a company with a strong continuous-delivery mindset, wanted to improve the situation and informally adopted the pipeline. As a developer, she did not have permissions to change the QA stages of the pipeline herself, so she tracked down the QA team members who had originally worked with the team.

Two QA engineers in particular were happy to see their original efforts restored to working condition, but with a backlog of QA work for other projects, they were only able to help sporadically.

None of the required fixes was complicated in itself – archiving of test results to a wiki had broken due to a change in API, the location for automated publishing of documentation had moved, etc. – but without an official project and associated billing code, the QA manager was not willing to let his busy team take time to tackle these issues. In the end, only an escalation to the VP of engineering and continued efforts from the developer and QA team members finally led to approval of a pipeline maintenance project.

Such an ongoing project is now created as standard for continuous-delivery implementations at the company.

Conclusion

Driven by market and customer pressure, market leaders have implemented continuous delivery as a strategy to accelerate time-to-market, and leading organizations are looking to follow. While much has been published on the principles and processes of continuous delivery, practical advice on how to approach and plan a continuous-delivery implementation in an enterprise environment is hard to come by.

Analyzing which of the common challenges to continuous delivery apply in your situation should be a preparatory step in your implementation. Mitigating any challenges that you identify early in the project cycle should help your implementation progress smoothly. Experience in addressing these issues will also prepare you to effectively address the next set of challenges you will encounter as the implementation progresses.

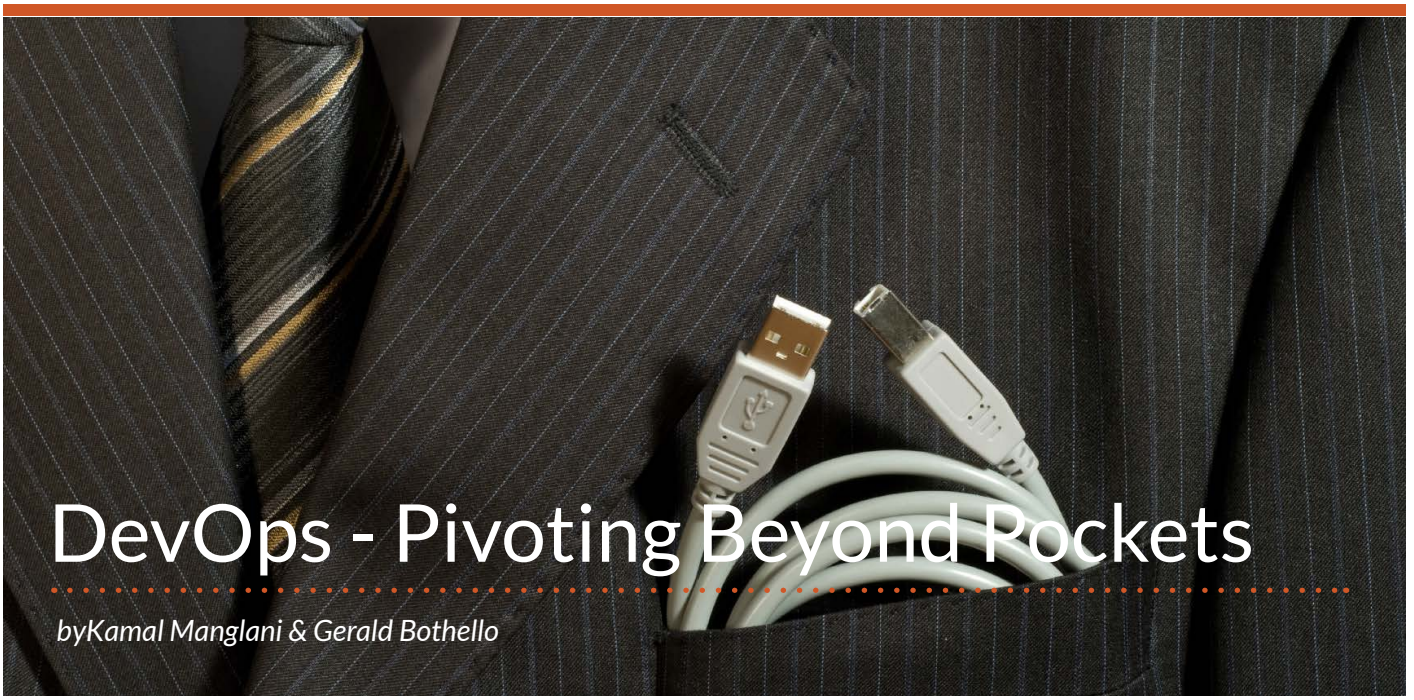
Gaining an accurate picture of your current baseline and structuring your implementation in measurable phases are the first steps to clearing the way for your first delivery pipelines with defined roles and responsibilities for each phase.

Your continuous-delivery implementation will then be on the way to providing faster releases, more reliable feature delivery, and steady improvement driven by quicker feedback and better insight.

ABOUT THE AUTHOR

Andrew Phillips is VP of products for XebiaLabs, providers of application-delivery automation solutions. Andrew is a cloud, service delivery, and automation expert and has been part of the shift to more automated application-delivery platforms. In his spare time as a developer, he worked on Multiverse, the open-source STM implementation, contributes to Apache jclouds, the leading cloud library, and co-maintains the Scala Puzzlers site.

READ THIS ARTICLE ONLINE ON InfoQ



DevOps - Pivoting Beyond Pockets

byKamal Manglani & Gerald Bothello

Today, organizations that are on a DevOps journey often end up implementing DevOps in pockets that are not scalable. The transformation to a DevOps-centric culture can hit or miss with this approach. This article offers a summary of traps and tips to consider.

Traditional infrastructure operations roles are no longer scalable. The traditional system admin and engineering roles such as network engineer and storage engineer are rapidly changing. The difference between a developer and operations engineer is becoming more and more invisible and will eventually dissolve. This is part of a massive shift in the IT infrastructure called "DevOps".

As Dr. Ahmed Sidky puts it: "You can't buy a culture transformation. It is hard work from within the organization". An uncalculated and solely engineering-based approach to DevOps can be less effective and unsustainable. Operations are generally locked in a fixed mindset with focus on control (controlling change, controlling risk, etc.) They often harden their controls under the delusion that they can actually control change and risk. The more agile the dev mindset is (and it's always more agile than the ops mindset), the more tension and friction it produces.

Let's start with potential traps:

1. Lack of clear vision on outcomes: Right from the C-level executives, there must be clear vision and outcomes expected from a DevOps transformation. The initiative must be treated as a major program with high visibility in the organization. Pivoting to a DevOps-centric enterprise is not a trivial activity. It involves a culture change and will need all hands on deck.

2. Lack of a transformation roadmap: Design a transformation roadmap, taking input from all impacted teams and not from individuals. Alignment across all impacted groups is vital and a critical success factor. For example, ignoring product management while building a strategic roadmap could become a major problem during execution.

3. Underestimating the scale of change and thinking of it only as a tooling or procedural change: The transformation needs hands-on technical expertise rather than PowerPoint decks. This is important from the VP and below. Core enterprise and solution-architecture teams must exhibit empathy along with the right skill set. Your architects must be terrific communicators, must have hands-on coding skills, and must blog regularly!

4. Lack of management empathy: The do-ers and executives must both have great listening and

collaboration skills. This also implies letting teams form and empowering them.

5. Lack of baseline measurement, clear definitions of done, and ongoing metrics for learning throughout the DevOps journey: Two of the easiest metrics to implement are customer satisfaction and cycle time. Some organizations may want to use cost of delay to rank DevOps priorities.

6. Rigid process framework: This creates no value but needs to be followed for the sake of documentation or the delusion of control.

7. Operations-change management: Extensive approval cycles for operations changes and an inability to make the bottlenecks visible create waste and negative value.

8. Lack of agile and lean education for both leadership and staff: Teach the culture. Taking an individual approach to training creates potentially little value and may even be detrimental to rapid team storming, norming, and forming.

Simply put, change management is hard!

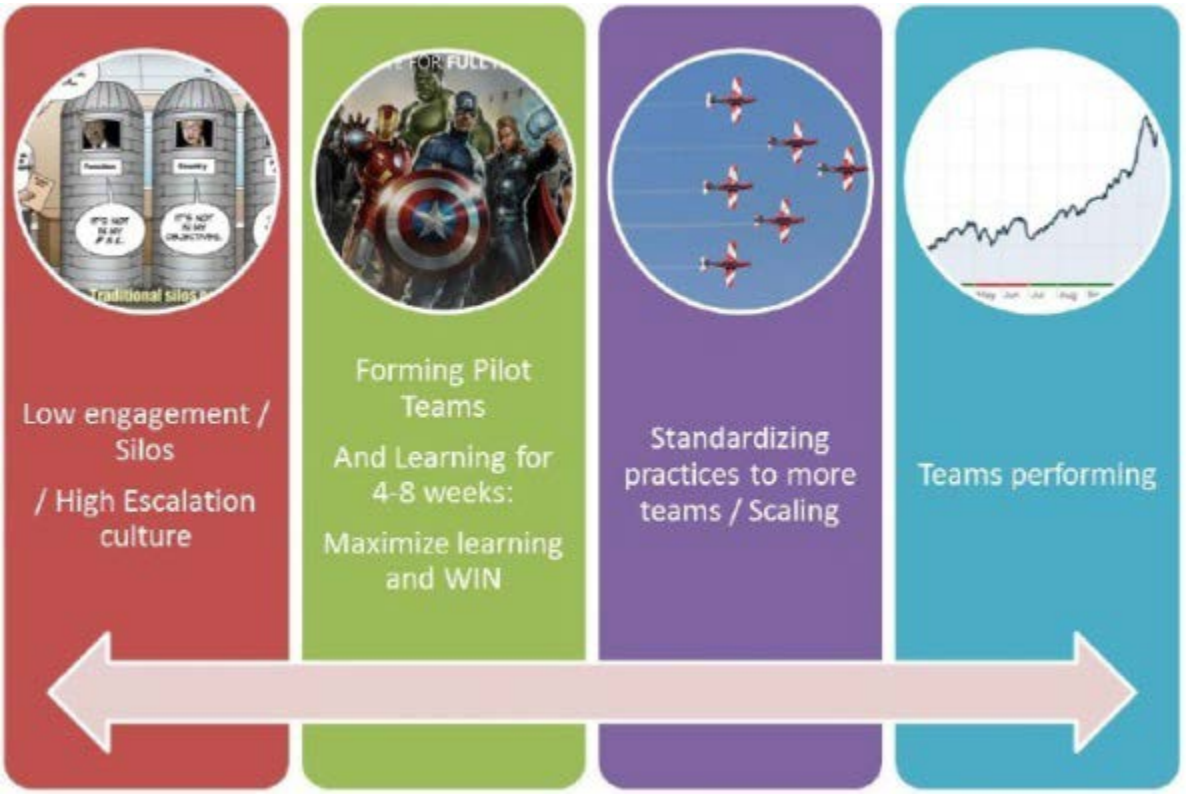
Recently, authors Gene Kim, Kevin Behr, and George Spafford developed a fiction novel called *The Phoenix Project* that outlines what a hypothetical DevOps transformation looks like. While the novel is a way for people to understand how DevOps change can happen, it might be great to hear about some checks and balances that go into this transformation. Here is a simple roadmap framework that may be customized to suit your organization's needs.

Based on the foundational vision for a DevOps transformation and the gap to be bridged in organizational culture, the organization may or may not be able to make the change sustainable.

Here is how we suggest pivoting to and implementing DevOps, based on our experience:

1. The leadership team must be ready for a **real** change in infrastructure operations and ensure organizational awareness that ops is an **internal customer** to dev, portfolio management, and product management. Organizations need to realize that DevOps is where the value gets realized. Think of it as ValueOps, not just DevOps.

2. The leadership must be willing to make **hard decisions** in staffing, in restructuring the organization to get rid of silos, and in rethinking



major strategic vendor contracts if needed. Some of the strategic vendors that came with the era of mass outsourcing may not be compatible with or sufficiently flexible to operate in a more agile environment.

3. Remove excessive approval gates from heavily bureaucratic change-management meetings.

4. Invest in **automation tools** and **role transformation**. Engineers' hardware skills need to be supplemented with crosstraining on technology. This takes care of situations in which teams wait on an individual to carry out task that could be performed by others.

5. Form **communities** to share the emerging good practices that can be leveraged across the organization.

6. Get ready to **restructure** entire departments. Once the agile teams are formed (scrum/kanban), the waste will be exposed, which will bring about transparency in stories/tasks delivered by the various teams. This cycle will further expose and elevate opportunities for improvement such as speeding up delivery.

7. Invest in developing **technical skills**, including three-dimensional thought process and the ability to pair quickly with application-development teams. The emphasis needs to be on developing talent in people rather than on process. Organizations should experiment with rotating people through a variety of roles so they develop greater empathy and experience.

8. Bring in code-quality profiling and test automation. Tools for configuration management and code quality will not provide full benefit if the code base is not checked for quality based on **standard quality profiles**. The quality profiles must be part of the performance-management system as a shared goal and code quality should be analyzed with quality thresholds that will reject poor code (eliminating technical debt at the source). Make it difficult for bad coding practices to thrive.

9. Get ready to change the **performance-review** process and focus it more on teams rather than individual performance.

10. Most of all, start with a team that is open to **embracing change** and that will learn quickly in order to achieve success.

11. Ensure your organization has a decent portfolio-management framework that uses the unit of **capacity as team** rather than headcount. Portfolios should look at whole teams servicing a market/P&L and ideally these product-centric teams will remain intact beyond any single project. This is especially important when you are dealing with large scales. Enable and empower engineers on the team to pick up cross-functional skills; for example a sysadmin must be able to work on cloud computing as well as on physical hardware, and on various operating systems.

12. Developers and Infrastructure must start to speak the **same language** through tools. Have a strategy for the deployment pipeline. Don't just look at it as a code line-up but as a value stream right from product management to deployment. Ensure that every team and department manages a high-quality deployment pipeline. Every time there is a code failure, it must be fixed, validated, and deployed in the respective environments. This will enable you to keep all your environments in sync and updated with code/bug fixes.

13. Reduce batch size to reduce deployment risk and to improve overall cycle time. This depends on effective collaboration between engineering and product management. Define work packages as weekly value increments.

14. Optimize teams based on value streams. Optimize product owners, too, within the DevOps teams. Stretch the definition of the DevOps term to fit product owners; without them, it is just not effective enough.

15. Finally, make work visible through teams. DevOps needs standups not within infrastructure but within the value stream. It is tricky to give status to every development team. A simple visual board that reflects status as backlog/in-progress (WIP limit)/review/done will go a long way toward improving customer satisfaction and visibility.

Summary

Don't be afraid to fail fast to succeed sooner, and don't fear adapting or replacing your traditional infrastructure management as it may be causing damage rather than fostering a better-performing organization. The traditional infrastructure and application-developer roles are changing. It is time we adapt to the benefits of DevOps and scale it beyond the pockets to the top of the value stream, starting with the product owner.

Acknowledgements

As members of the broader DevOps community, we would like to acknowledge the thought leaders whose strategic direction, books, and articles have helped our journey.

ABOUT THE AUTHORS

Kamal Manglani is an internal agile coach with WalmartLabs. As a strong, hands-on practitioner, he has delivered cutting-edge technology products to fast-paced Fortune 500 companies and has successfully implemented agile across global brands in the US, European, and Indian markets. Kamal has pioneered and customized agile practices within the IT infrastructure portfolio through the application of lean kanban principles. He regularly coaches agile in non-IT functional areas such as HR and finance.

Gerald Bothello manages infrastructure delivery at WalmartLabs. He has over 14 years of experience in IT as a developer, architect, technical program manager, and manager in software and Infrastructure. He has successfully delivered several large-scale projects using cutting-edge technology and incorporating agile methodology to Fortune 500 corporations.

READ THIS ARTICLE ONLINE ON InfoQ

DevOps – The Reluctant Change Agent's Guide

by John Clapham

Agile and DevOps people talk a lot about ***what*** to do when they want to encourage change – "Form a self-organizing team", "Break down silos", "Automate everything". I note that people talk a lot less about ***how*** to achieve these good things, the enablers. Frustratingly, logical arguments don't always win, that's when it's good to possess a few of the change agent's skills.

I've had a few skirmishes with organizational change, MixRadioevolved from long release cycles to (almost) Continuous Delivery in 2011. During the process I learnt an awful lot, some of which I shared in a recent DevOps Summit talk. Firstly, I think a little theory helps conversations:

1. Don't Learn Everything the hard way – DevOps is a relative new comer, other groups have way more practice at making change happen. It's well worth learning from them, including Scrum (especially types of resistor), read The Goal or follow Deming and Kotter's work.

2. Learn how far (and fast) you can push change – Everyone has a different reaction to change, and often it is fine grained and unpredictable. When introducing something consider Viney's J curve - how much short term slow down and instability are people willing to accept?

3. Not everyone thinks like you – Remember those logical arguments? Everyone is wired differently, their logic circuits will inevitably output something different and unexpected. Consider how people will react, their motivations and values. If profiling sounds tricky at least clear your preconceptions and maintain an open mind.

4. t's all about starting a movement – That's the process of gathering a group and nurturing followers so the group, and idea, grows. Luckily it takes just three minutes of TED talk to explain.

Following up the theory are a few practical points:

People with influence can help create that movement, but position in an organization chart isn't necessarily a good predictor of influence, it's worth looking for the intersection between the chart and social network.

Be sensitive to "change disparity", where there is a wide gap in understanding,or approach, between early and later adopters. This can be as demoralizing and damaging as any organizational silo.

Feedback and dissent are vital to fuel ideas, and determine if you're on a productive track. Pressure, history and trolls inevitably bring negative

comments. Filter out those unproductive phrases, often at the core is a useful comment.

Role model – Setting a good example is a simple idea, but often the hardest to do, we all have off days. If you're advocating something be consistent, authentic and a promoter, trust in you and the idea are tightly coupled.

I believe there is a lot individuals can do to drive change, with or without permission, support from leadership or a change program. Of course it takes time, and starts small, but those immediate relationships are probably the ones with the most bearing on productivity and sanity. It's useful to learn the theories, ultimately bringing change is about earning trust, listening and understanding.

ABOUT THE AUTHOR

John Clapham is a Software Development Manager (or Agile Coach if you prefer) with MixRadio, based in Bristol. Working for companies like BT, HP, Nokia and Microsoft, not to mention the occasional startup, he's been leading teams and using open source tech for over a decade.

John is passionate about agile, coaching and finding ways to help teams to build great products. He's co-organiser of the Bath Scrum User Group and one ofthe creators of Experience DevOps. Very, very occasionally he blogs, and may be found as @johnC_bristol.

John has two rumbustious children leaving little time for interests other than coffee, lego, pies and hiding.

Is the Enterprise Ready for DevOps?

As the DevOps movement gains popularity, enterprises have started to adopt its concepts and tools to manage large infrastructures and complex delivery processes.

InfoQ asked some experienced DevOps adopters about the organizational and technical obstacles the movement needs to overcome to step into the enterprise world.

The panelists

Gene Kim - Founder of Tripwire and author of the upcoming book The DevOps Cookbook from IT Revolution Press.

Jez Humble - Consultant at ThoughtWorks Studios and co-author of Continuous Delivery, who blogs here.

Patrick Debois - Developer, manager, sysadmin, and tester who coined the term DevOps and organizes regular DevOps Days.

John Allspaw - SVP tech ops at Etsy.com and culture hacker.

Mitchell Hashimoto - Creator of Vagrant, operations engineer at Kiip, and open-source enthusiast.

InfoQ: To what extent does the size of an organization affect the successful implementation of a DevOps culture?

Gene: Being in a large enterprise IT organization is no excuse not to adopt DevOps types of practices. But it would be foolish to ignore the challenges caused by the cultural aspects and effects of incentive structures that happen in larger organizations.

There's a considerable amount of literature that the amount of counter-productive "us vs. them" tribal behaviors occurs more in larger organizations. It isn't restricted to dev vs. ops, but includes production vs. sales, marketing vs. development, development vs. QA, etc.

Dr. Eliyahu Goldratt described this as the "local optima" problem in his theory of constraints and his book The Goal. Often, the tribal warfare between groups is codified in their measurements.

For example, in manufacturing organizations, the VP of sales often would, in order to protect customer commitments, be incentivized to carry as

much inventory as possible; while the VP of plant operations, in order to reduce costs, is incentivized to reduce the amount of inventory. So, no wonder they're often warring with each other.

The classic DevOps example is the VP of development who is incentivized by feature time-to-market and thus is motivated to deploy as many changes as possible, but the VP of IT operations is incentivized by operational availability and thus is motivated to deploy as few changes as possible.

Overcoming this core, chronic conflict between development and IT operations is a key part of a DevOps transformation. When organizations make this breakthrough, the results are amazing. I've seen organizations with thousands of developers and hundreds of IT operations staff make incredible improvements in feature flow while increasing stability and reliability.

Jez: My personal definition of DevOps is "a placeholder for a continuing discussion about how everyone involved can work together to improve the business of building and running systems." So I'm going to argue that you can never be "done" with a DevOps implementation - as Mike Orzen said to me, you shouldn't think of such an implementation as a project or program because they have end-dates. Thus I believe DevOps should be implemented through continuous improvement - find out what your biggest problems are, try and define some quantitative goals you want to achieve, and then come up with some ideas about how you might achieve them. Run some experiments with some teams who have a high level of maturity and are willing to try things out. Then take the lessons learned from these experiments and apply them to other parts of the organization.

How does the size of the organization affect this process? First, of course it will take longer. Second, you have to get buy-in from the leadership to get the time and resources you need to carry these experiments out and accept that they will slow things down and cause pain in the short to medium term. Finally you need people on the ground who are willing to make the time to experiment and collaborate.

Thus in a sense a "DevOps culture" means an organizational culture in which continuous improvement is actually possible - one that fulfills the criteria above. That's something that's very hard to implement, and usually the biggest barrier in large organizations is that the leadership - perhaps unwittingly - doesn't understand the importance of creating such a culture, or know how to do it (see my discussion on Theory X and Theory Y below).

In theory, I don't think that the size of an organization should matter. In practice, I wouldn't be surprised if success on that front had an inversely proportional relationship to size, mostly because larger organizations have some characteristics that allow for scale, but may prevent large shifts in culture to propagate. For example, a company whose development group is geographically separate (different building, different continent) from the operations group are going to have a tough time getting empathy, cooperation, and frequent communication put together unless it invests in expanding how the groups communicate.

Having said that, if a company has small pockets of dev and ops groups co-located or who can communicate freely and given a wide charter on implementation of technology, I could see it working well. This is what's been suggested before, a skunkworks approach: start with a small group; get them to ship something that has higher quality and operability than the status quo; and then point to the cooperation/collaboration as something to replicate elsewhere. I've seen and heard about this approach working, but I haven't yet see it spread entirely across the org until it's the predominant cultural norm.

Mitchell: I've never worked in an organization with more than 50 people, at least as an engineer, so I don't have any experience to back up any answer to this question. It's always easier to invoke change in a smaller group than it is in a larger group. However, some large organizations are set up in such a way that incorporating a DevOps culture is less painful. For example, if the organization is broken up into several small, autonomous, agile groups of a handful of people, then incorporating DevOps into one group is almost as simple as doing so in a start-up environment.

When more people are involved, it is important to show each person the value that DevOps brings to any organization. For developers, this means shipping features faster; for operations, this means sharing more responsibility and getting more eyes

on the infrastructure. For business folks, this means improved overall agility, which is typically cheaper.

Patrick: As with any change, the number of people to be changed will affect the speed of change and impact: within smaller organizations, new ideas will take less time to reach all people and when the ideas reach people, the initial idea will stay more pure because it has not been translated/transformed along the way. This is regardless of top-down of bottom-up approaches.

Larger organizations will by nature have more structure/rules in place and changing them will also take more time, much like turning an oil tanker into a kayak. Still, the size is not a guarantee of success; even within small organizations, early failures or misinterpretations can lead to non-adoption.

Therefore I think it's important, as with all change, that there is value and benefit for doing so. Showing early and repeated success is crucial.

InfoQ: When talking about DevOps in large traditional organizations (enterprises), one of the main roadblocks seems to be the environment's heterogeneity (machines/OS) and the fact that existing tools seem to work really well only in particular environments rather than mixed ones. Do you agree?

Gene: Heterogeneity is a fact of life, and I don't believe it's actually a valid DevOps objection. After all, any IT-operations organization that has had any amount of success will struggle with complexity, where a sprawling number of platforms and applications starts becoming a drag on daily operations. When this happens, organizations need an architecture and platform strategy to help to decide where the organization will invest to create mastery, and which ones to abandon.

Overcoming complexity is part of the growing pains in any IT organization, with or without DevOps. In fact, done right, DevOps can help ensure that there's always fast flow of planned work, which should help accelerate the paying down of technical debt.

Jez: I actually think that the organizational issues I discussed in the previous answer are a lot harder to solve than the technical ones described in this question. The main roadblocks I find are concerns around governance (Can we have developers and operations people collaborate and still be SOX or PCI-DSS compliant?), empire-building by managers of functional silos, and an unwillingness to create slack time to work on improving the delivery system. This last problem exhibits itself in different ways in different parts of the organization. In development, you find managers who won't let developers do anything other than work on features and fix bugs. In QA, testers are focused primarily on performing manual testing. Operations spends its entire time firefighting. Nobody has time to think about improving the process. It's like a woodcutter who is under a tight deadline to cut down a lot of trees, and thus doesn't want to stop cutting down trees in order to sharpen his axe.

Of course, there are technical issues too. But more than environmental heterogeneity, I find that often a bigger problem is enterprise architecture. In enterprises, systems are usually big, monolithic codebases that are tightly coupled to each other and to COTS that has been heavily customized. Thus you can't deploy anything without deploying everything, which makes provisioning production-like environments for integration and testing purposes incredibly slow and expensive. The first priority here is to work on making their systems more modular so you can deploy individual modules independently in an automated fashion and find the majority of bugs through acceptance testing in a non-integrated environment. Modular architectures also make integration testing easier. When enterprise architects think about requirements for their systems, deployability and ease of testing and integration don't feature nearly as highly as they should.

John: I can see how someone might take that stance, but I don't think that it's a true impediment. I can't say with a straight face that the converse is true, that homogeneous environments would have a better time getting development and operations cooperating and collaborating. I guess one could say that the more homogeneous an environment is, the fewer automation (imaging, config management, deployment, monitoring, etc.) tools would be needed, which would make it easier for multiple parties to share experiences, code, and adjustments.

The ironic thing that I've found in companies that have a huge number of software stacks across pockets of the company is usually indicative of

historical rationales that follow axioms like "not invented here", "right tool for the job", and "my manager knows this tech, so we'll use this in our group", which are local optimums (good for the group), but globally sub-optimal (not good for the entire company). Part of a mature dev and ops relationship is to make tradeoffs explicit (in this case, when choosing software), and globally optimal solutions come from that.

In the extreme case where many groups all over a company have decided on completely unique software stacks, there's a danger that even if there aren't organizational silos, there can be implicit ones aligned with the stacks: for example, the PostgreSQL group, the MySQL Server group, the Windows group, etc. Walls that align on those boundaries are still walls to hurdle.

Mitchell: Heterogeneous environments certainly raise the difficulty level substantially. It is easy for many people to disregard this and yell "use the same machines and operating systems, obviously!" But I think most organizations would prefer a homogenous environment, and simply end up heterogeneous due to factors they can't control. Some software only runs on some operating systems, some parts of the organization move forward with new systems while legacy systems retain older versions, etc.

For myself, I can say that it is a top priority for my own tool, Vagrant, to support as many environments as possible. If Vagrant doesn't work properly in one environment, then it is a bug. As Vagrant is a relatively new tool and only recently starting to breach larger businesses, I'd say that as tools become enterprise-ready, one of the necessary core features is widespread system support.

Therefore, while heterogeneous environments are certainly annoying to deal with, I expect newer tools to embrace this sort of chaos.

Patrick: If you think of DevOps through the CAMS (culture, automation, measure, and share) model, I don't see any inherent DevOps issue, but more technological challenges. We should still use the best tool for the job, regardless. Reducing the number of tools will contribute to speeding up the learning curve. But then again, it's in the nature of enterprises to have a multitude of different apps build at different times with different technologies. It's important that you keep fighting this heterogeneity as a form of technical debt, especially when systems become legacy. I would not consider it a roadblock of DevOps adoption, though.

InfoQ: Opscode Chef's CTO Chris Brown recently mentioned the importance of migrating private Chef to a database (MySQL and PostgreSQL) more familiar to enterprise folks. Do you think avoiding technical disruption is a requirement for DevOps to thrive in enterprise settings? Or, on the contrary, is such disruption needed to trigger or reinforce cultural change within organizations?

Gene: I think the transition Chris Brown is referring to is not DevOps-specific, but more about how technology vendors need to fit within the technology capabilities of their customers. We went through a similar transition when I was the CTO of Tripwire. We migrated to Oracle and MySQL so that creating a new database instance would be a routine task (i.e. opening a ticket), instead of a novel one-off task involving days of research for our customers.

This reinforces the adage that initiatives such as DevOps should leverage existing processes and tools as much as possible.

Jez: You need to work out what battles to fight. For me, when people within organizations are trying to change the way they work, the key thing is to demonstrate measurable improvement as rapidly as possible, and certainly within a few months, and then keep doing it. That necessarily limits the scope of what changes you should embark on. The changes with the biggest lead times are political, so if you're going to avoid a big fight with the operations folk by agreeing to use Postgres or MySQL, that's probably a smart move.

John: I don't think that avoiding technical disruption is a requirement. Instead, I'd say that a requirement be that an organization have the ability to make tradeoffs in every technical choice explicit, in terms of performance, feature set, and operability. I see where and how Chris would say that, related to my comment in the previous question.

Now, it's probable that the cultural shift needs a big-bang moment, such as a large technical decision, in order to harness the enthusiasm and self-selection of engineers who are up for such a change. But is it a requirement? I'm not entirely sure.

Mitchell: It matters. Opscode Chef is open-source software, an open box that they expect organizations to be able to install and manage on their own. If an organization doesn't feel comfortable with this, they can go with private Chef. On the other hand, when something is a black box, technology choices do not matter so long as the company behind the software makes service-level guarantees about the software they are shipping.

Patrick: The balance is somewhere in between: to have people adopt new technology, they have to be able to relate to either the problem, technology, or solution you propose. Disruption brings people out of their comfort zones. Some people like that as a way of being challenged; others not. It's important you understand your audience. If, like Chris, you want to drive adoption of your tool, make the barrier for adoption lower by reducing the learning curve.

InfoQ: What would you say are the main obstacles and future for enterprises that adopt DevOps as part of their ALM?

Gene: Based on the work I've seen presented by enterprise IT organizations (e.g. Unum, Paychex, BNY Mellon, World Bank, etc.), DevOps initiatives are often pushed by the same groups that drive enterprise ALM and SOA initiatives. This makes sense. These are the groups motivated to improve service quality, and routinely work with enterprise architecture, development, QA, and operations.

Their challenge is to get all these different groups to act as one super-tribe, trying to solve a larger systemic problem as opposed to allowing these groups to act as warring tribes.

See the next question below for how ALM groups are uniquely suited to solve this organizational silo problem.

Jez: First you have to recognize that you have problems and be committed to solving them on an ongoing basis, and that doing this is going to be painful, especially in the short term. The scope of DevOps adoption is pretty wide, ranging from how you approach compliance and auditing to the architecture of your systems. People will need training, coaching, and support in everything from maintainable test automation to how to communicate effectively with other people in the organization. Finally, everybody needs a clear, shared vision of where they want to go in order to create a plan on how to get there.

John: I would say that organizations that don't view technology as a competitive advantage are the largest obstacles. This is my own bias, but when I see a company refer as a whole to the people who run the computers as the "IT department", as opposed to "application development" or "technical operations", etc., I see a company who very much views computers as a necessary evil for the business, not a vehicle for enablement.

I once worked at an advertising firm in Boston. The resourcing, focus, and ability to get things done in technology to enable the business was generally on par with the facilities group, which was charged with making sure light bulbs got replaced when they were burned out. The engineering groups were not seen as enablers for the business so therefore were left out of the loop of conversations about future direction and other ways that technology might help move the company forward. This meant that culturally, the engineering groups were treated as second-class citizens in the company, and therefore didn't get as much attention paid to progressive topics, which is needed.

So the greatest obstacle for enterprise companies is to get past the stereotypical view of IT as technicians keeping the servers working and start viewing them as engineering-driven groups that can bring a competitive advantage to the business, and therefore worthy of investing time and effort.

Mitchell: The biggest obstacle at this point is education. DevOps has been a turbulent movement the past few years because it hasn't been strongly defined. I think we're finally reaching a place where more people can comfortable say what DevOps is along with case studies showing it actually works. Given this, the resources are still too hard to find on what DevOps is and how you can transition your organization to a DevOps culture.

I'd like to see less hand-wavy talks on DevOps and more concrete step-by-step approaches to implementing DevOps. I realize that these approaches don't work for every organization, but at least it'll help guide folks.

Patrick: If you consider the challenges for the DevOps movement, it's to bring out more enterprise success stories compared to the many startup/ Web 2.0 company successes. People are trying with varying success, but they are often scared to bring out the non-successes, as they would consider them failures. Sharing failure is an important thing of learning, and needs to be stimulated.

For people within the enterprise, I think challenges are similar to those of any change being introduced (like agile several years ago). Depending on whom you talk to – CEO, CIO, project manager, or technical person ¬– it needs to be compelling and bring value. The obstacle for adoption is often the inertia of companies and looking wearily at the new trends. They need to be proven. People and their habits are really the biggest challenge. I found that often it becomes a belief question: does collaboration and shared goals increase value or do you believe stricter and clearer separation of duties have a better impact. Depending on the team and person's vision, they will be inclined to go for one or both. I even believe that some people will not thrive within a DevOps mindset if that's your core belief.

InfoQ: Do you think it's possible for large enterprises suffering from organizational silos to realize the benefits of a DevOps approach? How?

Gene: a. Recognition of a shared problem

What I've found in over five years of research is that the organizational-silo problem leads to an adversarial relationship between development and IT operations, which results in an ever-increasing amount of technical debt that slows down the rate of feature release and time to market, as well as increasing instability and chaos in the IT production environment.

The challenge is always how to show development and IT-operations leadership that neither of their goals are being met and, left unaddressed, will continually get worse.

b. Willingness of management and practitioners to face it

I've found that when we demonstrate to leadership how the entire organization is jeopardized and threatened by this inter-tribal warfare, they will act decisively to end it, working together to overcome big challenges that could scarcely be imagined in months prior.

Outcomes include shorter deployment times, better automated testing, less time spent doing code packaging, more successful releases, and faster deployment cadences. Furthermore, with better feedback loops between IT operations and development, developers get needed information faster (often through self-service), the numbers of escalations drop because developers have cross-trained IT-operations staff, and a higher percentage of organizational cycles are spent paying down technical debt.

The result is faster flow of planned work, while increasing the stability, reliability, resiliency, and security of the production environment.

And when that happens, development and IT operations are truly helping the business win.

Jez: Yes. For me, the most important message of DevOps is that nobody is powerless – there are little things any of us can do every day to make things better. Developers can go and have lunch with the DBAs and find out why they hate the developers, and what they could do to make things a little bit better. People in operations can carve out 30 minutes of their day to set up version control and check all their scripts into them so that they can improve their configuration management and give read access to developers so they can understand how production environments are created and maintained. Set up cross-silo lunch-and-learn sessions to share knowledge and develop relationships between groups. Create an internal blog. Nobody should be waiting for a mandate from on high to start up some kind of "DevOps program".

John: I do, but only by investigating the true reasoning behind the original silo-ing of the groups. If you can't illustrate quickly and easily why it's beneficial to have groups cooperate, then you can't make the case for doing it. Also, if upper management isn't on board, especially management that believes that they are benefitting from silo-ing the groups (vis-à-vis, a convenient "cover your ass" and finger-pointing boundary), then traction will be tough.

I believe the trick is to convince plainly that the effort put in to cooperate, collaborate, and communicate

between dev and ops is an investment worth making, worth more than the cultural security blanket of keeping a finger-pointy culture. This persuasion can't happen unless both teams recognize the legitimate place for domain expertise, which means that they defer to each other along those boundaries.

Mitchell: I think it is possible. The people in charge of these silos need to see the benefit DevOps has on the business and the culture. A change for the sake of change is pointless, but when there is actual value gained, then we see change happen.

Patrick: Sure, they can and I would even say they probably often need it the most, as the size of company often is driver in creating groups of people as a natural human reflex for grouping similar things. The approach from switching from project teams to product teams with full responsibility from beginning to the end can work well within an enterprise. This will foster collaboration and better results. But much depends on the culture of the company: IT doesn't live in a vacuum, it's a balanced act. Those product teams often seem to be a duplication of effort/costs by less leveraging of the so-called economies of scale. Still, the benefit from adding agility and self-management outweighs this.

InfoQ: DevOps appeared a year ago on the Gartner hype cycle as an emerging approach holding less than 1% enterprise market penetration but expected to reach its plateau of productivity in five to 10 years. Does this match your own expectations?

Gene: I believe we're only at the very beginning of DevOps adoption, which will result in a massive increase of productivity in enterprise IT. When I've asked enterprises who've adopted DevOps how to respond to people who say DevOps isn't for enterprises, they're always puzzled. To them, it's merely a prioritization issue.

I believe that the DevOps patterns are the inevitable outcome when you apply lean principles to the IT value stream. As Dr. Deming once said, "Survival isn't mandatory."

Jez: I'd be surprised and very happy if we manage any kind of pervasive change in five to 10 years. What has depressed me the most while working on my new book on implementing continuous delivery is the extent to which people have been talking about these ideas for decades. Peter Drucker was discussing the differences between knowledge workers and manufacturing in the late '50s, and we still haven't learned those lessons. For example, he said, "Productivity of the knowledge worker is not – at least not primarily ¬– a matter of the quantity of output," and yet you still find many organizations that measure developer productivity in terms of lines of code per day. Lines of code are a liability, not an asset! Douglas McGregor came up with Theory X and Theory Y in the 1960s, and it is clear that knowledge workers can only be effective in an organization in which managers have a Theory Y attitude, but that is certainly not the norm in many (perhaps most) enterprises today.

John: Frankly, I don't pay much attention to Gartner, and I'm not really interested in the business of industry predictions. If an analyst is right about a prediction, it didn't really move the industry forward. If they're wrong about a prediction, it didn't move the industry forward. What I'm interested in is the dirty details that bring maturity to an organization's ability to engineer solutions to problems in the present.

Mitchell: Yes. I work in a reasonably young, "hip" environment, so DevOps has been easy to integrate. But it still has obvious rough edges. I'd say in five to 10 years it'll reach a solid stabilization point.

Patrick: I'm aware that many of the DevOps discussions are being done by the early adopters and will take time to spread. Some time ago, people questioned agile, now about 10 years old, and how much its market penetration was. I have no crystal ball, but I hope to think we still have a lot of growth potential. The technical aspects like infrastructure as code are faster because they are already instant practical. In the future, culture adoption will increase but will take longer.

InfoQ: Could too much hype eventually harm the movement by focusing on creating DevOps jobs without actually addressing collaboration problems between devs and ops?

Gene: Creating job descriptions with DevOps indicates that business leadership is seeing something of value, even if they can't say what the roles, responsibilities, or daily work are. It doesn't

invalidate the need. I think it's just indicative of how early we are in this journey.

Currently, DevOps is more like a philosophical movement, and not yet a precise collection of practices, descriptive or prescriptive (e.g., CMM-I, ITIL, etc.). The intent behind the The DevOps Cookbook project is to catalog what the high-performing DevOps organizations all have in common that result in their extraordinary performance outcomes. Our goal is to create the prescriptive guidance to create the culture, values, processes, procedures, and daily work so that we can repeatedly replicate their results.

You can learn more about this project here.

Jez: If organizations were looking for "DevOps people" to enable organizational change, I could get behind it. But many organizations are attempting to address the dev/ops divide by creating a new DevOps team whose job is to span the boundary between existing functional silos that aren't expected to change the way they work. This is, of course, deeply ironic. The focus of organizations interested in actually achieving the goals that DevOps enables – satisfying the customer through continuous delivery of valuable software, for example – should be helping their existing people change the way they work and learn DevOps skills such as how to treat infrastructure as code. The other problem with trying to hire in DevOps is that the demand for good people with the necessary experience vastly outstrips supply. And many of these people have been sucked up into organizations which then use them to try and mitigate the symptoms rather than address the root causes, which doesn't do anything to increase supply.

John: I think that too much hype can harm anything, eventually. Frankly, I'm seeing it already. Many reference DevOps and continuous delivery as synonymous. How frequently you deploy, how you deploy, and how automated you are has nothing to do with DevOps, other than organizations that do make liberal use of automation and continuous deployment are often known to have a DevOps culture.

I also think that in general, there is a tendency to minimize the "soft skills" in technology companies. I've seen projects where engineers will spend months trying to automate a process whose main benefit is that they don't have to talk to someone in another group every so often. This is ludicrous, and not a basis for a mature engineering organization.

So I do think that the hype can harm the concept, in the way that it could drive the focus away from the cultural focus and towards a "DevOps-in-a-Box" shrink-wrapped, technology-only concept.

Mitchell: We see this sort of issue already, where people call themselves "DevOps engineers" or say "We do DevOps." Again, I find this to be caused ultimately by a problem in education. Therefore, the hype surrounding DevOps gives off some false information, and the circle perpetuates.

Patrick: It's hard to help people that want to believe in fairy tales told by vendors, by which I'm not saying all vendors are bad. The hype I'm seeing comes from cranked-up people who have a financial incentive in doing so. The jury is out if they deliver their promises. The danger is that the so-called DevOps label can get a bad name due to this. One bad experience weighs more than many positive experiences. Again, if you want to think bad, you will have all the ammunition. By keeping the good content and stories coming we can keep the positive side on things. Similar things have happened to agile and lean, but the difference has been the speed at which DevOps as an idea has spread around the world. But that speed is also an advantage when getting new stories and ideas promoted or adjusting remarks. Learning from critics is important and we would clearly want to avoid to have DevOps people turn to fundamentalist views like "infrastructure as code" is better than shell scripting.

InfoQ: Most of you have participated at DevOps Days and other conferences in this field. How do you see the evolution of DevOps and in particular its expansion from Web-based agile companies to legacy-based process-oriented organizations?

Gene: I believe that we saw DevOps patterns first in the Web-based agile companies, because that's where competition was the fiercest and downtime and user-response times directly equated with revenue.

Because IT is critical to almost every business, we'll see it adopted next in the more traditional IT organizations. Mike Orzen and I did an article on value created by adopting DevOps. We believe it

will create $4 trillion of value worldwide each year, which is greater than the entire economic output of Germany.

That will have a huge positive impact on productivity and standards of living and reduce the amount of damage that working in IT often causes our fellow humans. All this is why I think DevOps is genuinely an important movement.

Jez: There are plenty of Web-based agile companies who don't have a DevOps culture – although you might argue with their definition of agile, since arguably if you were really doing agile you'd have a DevOps culture anyway. In my experience these companies face similar problems to organizations that don't define themselves as agile but are still interested in implementing DevOps. The main risk is that organizations try to adopt DevOps without a clear understanding of what they are trying to achieve, and in particular without defining measurable outcomes that they want to track to.

People have a tendency to start these kinds of initiatives by buying a bunch of tools, which is almost always a Very Bad Idea – tools are important, but they shouldn't be your starting point. A good way to start would be to look at the maturity of your existing capabilities in building and running software systems, and then how you'd like them to look based on the measurable outcomes you want to achieve, and then to put in place a plan to get from A to B in not more than a few months (if you think it's going to take longer than that, you need to choose a closer destination).

In my experience though, the main barriers to implementing the plan you come up with are politics, culture, architecture, bad tools, and a lack of skills.

John: I think there is a lot of cross-pollination that can still happen, but the empathy between Web companies and enterprise companies has to grow from where it is at the moment in order to truly move forward. I still see a good amount of condescension from both sides.

I've been told by a tech exec at a bank that Etsy (or any Web company) wasn't a "serious" endeavor, that his bank works with "serious money" which means they can't "screw around" like Web companies do. I've also seen Web companies poo-poo the enterprise because they're spoiled with their small user base and non-24x7 working environments.

Until there is a shared understanding between those groups, the healthy and mature swapping of ideas and concepts is going to be slow.

Mitchell: I'm quite new to the field, having only been involved for a year or so, so I'd feel more comfortable leaving this question to the more experienced people on this panel.

Patrick: Over the last years of DevOps, we have seen people initially adopt the automation practices. This is slowly shifting into the area of measuring and metrics on a more practical level. I tend to think of it as the return that feedback-channel automation requires: automating bad practices does not improve anything unless you learn of it. Now the security field is recognizing the value of DevOps and I'm excited to see people from different parts of the company adjusting their ideas. Legacy-based, process-oriented companies will only partly be affected of a lack of agile tooling but hugely by the aforementioned stubbornness and inertia of the human mindset.

InfoQ: Any last words about these topics?

John: I would just like to say that reports from practitioners in the field rule the day. DevOps as a concept or collected ideas will live or die by its ability to get things done for a business – which is why sharing these stories at conferences and in articles is so important. So for people who believe that collaboration, communication, and cooperation between dev and ops is a good idea: tell your story. And don't leave out the nasty bits. Tell it all.

For those others who aren't convinced of DevOps: if you think that your business will thrive further by motivating people to hoard information, isolate their functionality in the resource chain, or otherwise work at cross-purposes with other groups…go for it – as long as you share that story with the rest of your field.

Cultural shifts don't happen because people invent new words. They happen because people take action, and then tell stories about it.

Mitchell: Use Vagrant! I'm obviously biased but multiple sources have told me it helps drive DevOps

change much more smoothly throughout an organization since you can more freely experiment and play with the tools necessary to properly incorporate a DevOps culture.

Patrick: Sharing amongst peers, organizations, and industries is a crucial factor in the growth and richness of DevOps ideas. Similar to sharing code, I'd like to capture new ways of thinking and keep on encouraging people to do the same. DevOps has shown that grass-roots ideas can have an impact just like you can express yourself on YouTube vs. buying airtime on TV. We are the industry that we make it to be.

ABOUT THE PANELISTS

Patrick Debois has taken a habit of changing both his consultancy role and the domain in which he works in order to understand current IT organizations: sometimes as a developer, manager, sysadmin, tester, and even as the customer. During 15 years of consultancy, there is one thing that annoys him badly: it is the great divide between all these groups. But times are changing now. Being a player on the market requires you to get these "battles" under control between these silos. He first presented concepts on agile infrastructure at Agile 2008 in Toronto, and in 2009 he organized the first DevOps Days. Since then he has been promoting the notion of DevOps to exchange ideas between these groups and show how they can help each other to achieve better results in business. He is currently organizing DevOps Days events all over the world.

Jez Humble is a principal at ThoughtWorks Studios, and co-author of the Jolt Award-winning *Continuous Delivery*, published in Martin Fowler's Signature Series (Addison Wesley, 2010). He has worked with a variety of platforms and technologies, consulting for non-profits, telecoms, financial services, and online retail companies. His focus is on helping organizations deliver valuable, high-quality software frequently and reliably through implementing effective engineering practices.

READ THIS INTERVIEW ONLINE ON InfoQ

Gene Kim is a multiple-award-winning CTO, researcher, and author. He was founder and CTO of Tripwire for 13 years. He has written two books, including *The Visible Ops Handbook*, and is now writing *When IT Fails: A Business Novel* and *The DevOps Cookbook*. Gene is a huge fan of IT operations and how it can enable developers to maximize throughput of features from "code complete" to "in production" without causing chaos and disruption to the IT environment. He has worked with some of the top Internet companies on improving deployment flow and increasing the rigor around IT operational processes. In 2007, ComputerWorld added Gene to the "40 Innovative IT People Under the Age of 40" list, and was given the Outstanding Alumnus Award by the Department of Computer Sciences at Purdue University for achievement and leadership in the profession.

Mitchell Hashimoto is the creator of Vagrant and is an operations engineer for Kiip. He is passionate about all things ops and open source, and enjoys spending hours of his free time each day contributing to the community. In addition to simply contributing to open source, Mitchell enjoys speaking at conferences and user groups about Vagrant. Mitchell can be found on GitHub and Twitter as @mitchellh.

John Allspaw has worked in systems operations for over 14 years in biotech, government, and online media. He built the backing infrastructures at Salon, InfoWorld, Friendster, and Flickr. He is now VP of tech operations at Etsy, and is the author of *The Art of Capacity Planning*, published by O'Reilly.