ca technologies

DevOps Perspectives 4

Actionable guidance from DevOps practitioners and thought leaders

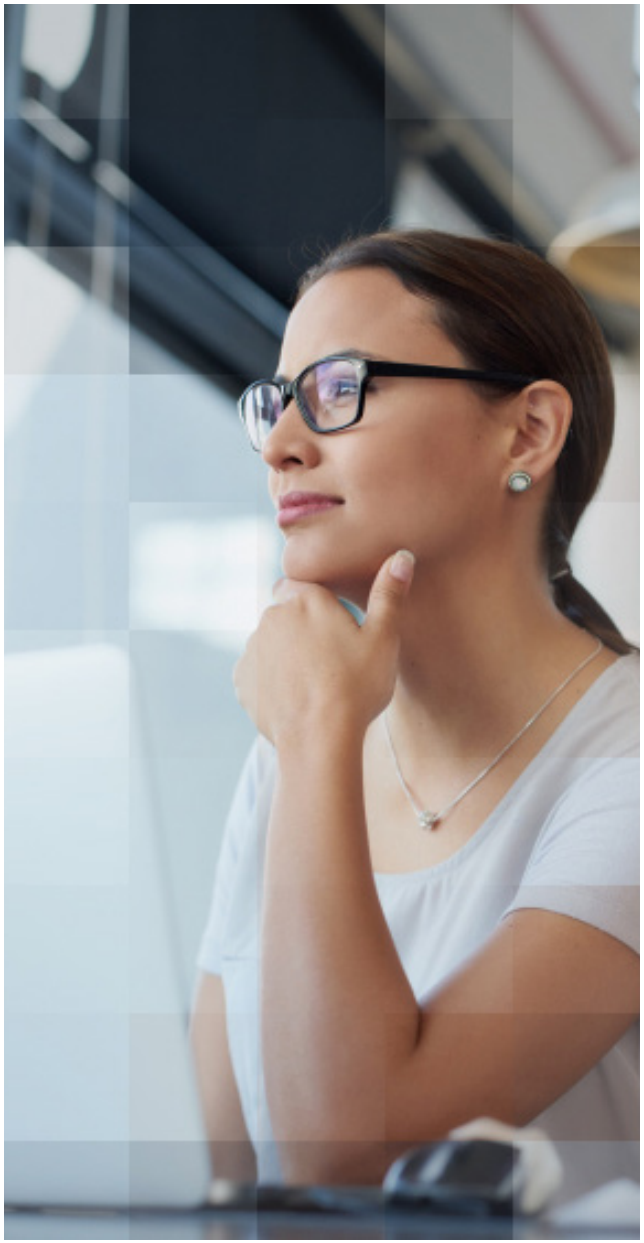
DevOps—
Carpe Diem Moment

It's indicative of the maturity of any movement that there comes a tipping point when it's time to think about the myths and misconceptions that have arisen during its development. DevOps has arrived at that moment.

While debate still goes on about best practices and processes and DevOps teams secure their place in the IT organizational hierarchy, there's danger ahead if certain misconceptions aren't stamped down upon, as we see most obviously in this issue's "Cargo Cult 101" feature. The dangers of allowing false premises to take hold can be seen elsewhere. There have been so many attempts to fire the proverbial silver bullets into the body of public sector IT over the years that it may be tempting for some to allocate DevOps to the role of the latest in a long line. But in fact, as we can see from the work of the likes of Digital Transformation Office (DTO) of the Australian government, which we feature later, there's a genuinely transformative role for DevOps to play if there is also understanding of its potential and its limitations.

As ever it's necessary to think in new ways about different ways of working. For new ideas, such as the code-driven threat modelling highlighted in this issue, to work, DevOps need to find their role in a wider ecosystem of responsibilities. There needs to be greater outreach to and collaboration with the likes of governance and compliance personnel, for example, not just the patient waiting for acceptance from the IT team at large.

Tipping point moments can be scary points in time. But they're also indicative of growing maturity. DevOps evangelists need to seize the day and step up to the opportunities presented.



CONTENTS

DevOps—Government's Neglected Digital Transformation Tool?

Lindsay Holmwood

Around the world, from country to country, one idea enjoys widespread acceptance—government IT has not delivered on its promise. It's that perception that has driven various governments and public sector organisations to invest in digital transformation programmes and cloud computing adoption.

DevOps should be playing its part in this wider citizen-centric services delivery revolution, but to date the public sector has not embraced the movement to its fullest potential. In fact, public sector organisations are a third less likely to be using DevOps than their private sector counterparts according to a study of 1300 private and public sector respondents carried out by Vanson Bourne on behalf of CA Technologies.

"The problem with most DevOps programmes is that they focus on just one aspect."

Yet of those public sector bodies that have embraced DevOps to a greater or lesser degree, there are demonstrable results. More than one-third (35 percent) reports an improved quality of deployed applications, while nearly half (49 percent) have seen a reduction in spend on development and operations. That latter point is particularly pertinent as governments around the globe strive to do more with less.

It is important of course to recognise that public sector organisations differ from private sector ones in a variety of ways, ranging from operational structures and drivers through to metrics and success criteria. For example, while a private sector business exists to create profit and serve its shareholders, public sector bodies are there to serve the citizen in the best way possible.

Government services need to run as smoothly as possible at all times as these are systems on which, quite literally, the lives of individual citizens may depend. So the public sector needs improved efficiencies and guaranteed delivery with the high stakes involved also resulting in a more risk-averse culture where bureaucracy becomes a barrier.

So, how can DevOps adoption be encouraged across the public sector? Lindsay Holmwood is infrastructure and platforms lead in the Australian government's Digital Transformation Office (DTO) and an advocate of the potential for DevOps in reforming and improving government IT service delivery.

"The way that IT services have been delivered by government have tended to be stratified and lagging behind what we've seen in the private sector," he says. "People in government are vaguely aware of the term 'DevOps,' but there's not a lot of understanding about what it means in practical terms.

"There are pockets of innovation throughout government, with people doing DevOps without actually realising it. There is a huge overlap in what the DevOps movement tries to achieve and what the agile movement has been trying to achieve. So you see departments and agencies trying to be agile and touching on DevOps at the same time.

"It's essential," says Holmwood, "that government organisations are aware that DevOps is not just about adopting some new technology, but rather about a fundamental shift in the way things are done.

"There's a big risk in government adoption that the term gets tainted by a lot of vendors selling supposed DevOps solutions to solve a so-called DevOps problem," he advises. "The real risk is that the message that ends up being pushed is 'buy this tool and you'll achieve DevOps.' What does that mean? That literally means nothing. It's like saying 'buy this agile tool and achieve agile.' It's meaningless.

"It's an alluring idea that you're going to use this set of tools and then you're going to reap all the benefits. That's a one-sided view of it," he adds. "But the DevOps movement is about two things. It's about the technology that's going to help, yes, but it's also about the culture and the need to change the culture within an organisation.

"The problem with most DevOps programmes is that they focus on just one aspect. You end up with a push built around technology which doesn't change the culture of delivery so you don't get the full benefits."

All of which raises the question of how to achieve that cultural shift. Public sector cloud computing adoption has been inhibited, particularly at local government level, by an inherent conservatism among risk-averse IT decision makers. "While various governments have attempted to drive up adoption through cloud first mandates, this 'do as we say' approach won't succeed for DevOps cultural acceptance," says Holmwood.

"You can't forcibly introduce DevOps into the public sector or you're going to end up in a situation of passive acceptance where people will say all the right words in public, then go back to their offices and do the exact opposite," he says. "You need to convince people of the value of changing the way that they work."

It's necessary to approach different parts of the organisation in different ways. "You have to craft your message to your audience. You need one message for the senior executive, one for your team leaders, one for the developers and so on," says Holmwood.

"If you're selling to the top, those people at the top are totally going to get the idea in business terms. They want to know about getting products to market more quickly and deliver services more effectively that are focused on user needs and are of a higher quality. That's a well-crafted message that's easy to communicate.

"But if you go straight to the top, then you can end up with a complete lack of interest underneath. You get people there saying it's just another flavour of the month thing. Again, they'll nod and agree, but go back to the office and do something else.

"There are multiple facets to encouraging adoption. It's not an either or choice," he adds. "If you try to have the grassroots of the organisation sell the idea of DevOps to the higher-ups, then that's going to be an uphill battle. You have this traditional hierarchy that says that the leader at the top is smart and the workers at the bottom are dumb. You can have people at the bottom saying 'The way we do this is wrong and we should do it another way.' But that can lead to pockets of resistance as those people bump up against the bureaucracy of how things have always been done until they get so fed up that they end up leaving.

"Taking a 'softly, softly' approach to introducing a DevOps culture is an idea that can be borrowed from the private sector," suggests Holmwood, pointing to what's happened in the financial services world. "You can see a two-track approach," he says. "In the slow track you have a bunch of stuff like your traditional back-end systems, the mainframe-based stuff, the really big enterprise applications. In the fast track, you have all the innovation happening, where organisations are using modern programming languages, continuous delivery and good modern DevOps practices. All the new products are going to be delivered on the fast track. Longer term, there will be a programme to migrate the heritage systems, the systems that are part of the reason that the big organisations actually exist."

The same can be applied to the public sector where the heritage systems will by necessity co-exist with new digital solutions for some time as part of a wider digital transformation movement that will see new ways of working emerge.

"There are multiple facets to encouraging adoption. It's not an either or choice"

"We're doing differently is that when we're building services for users to engage better with government online."

"The core thing we're doing differently is that when we're building services for users to engage better with government online, we're all working together, not pulling from a team of DevOps people or product managers or whoever, passing work between them," advises Holmwood. "Instead we have cross-functional teams. That's the aspirational goal of what DevOps is about—getting all the stakeholders in one place all together."

There is also, of course, a need to have the necessary talent in place to support these objectives. While much is written and spoken about governments having outsourced their talent base to third party suppliers over the years, Holmwood is optimistic that the situation isn't as bad as sometimes is painted.

"The skills are there, but it's talent that has yet to be unlocked," he says. "I'm very involved in the hiring process at the DTO, hiring across all the technical positions and that gives an interesting insight into the tech skills on offer. There are already people in government who have the attitude and understanding of the value of cross-functional teams. Our job is to help agencies develop their agile and DevOps capabilities. There is an enormous depth of talent waiting to be unleashed."

Putting the Security into **DevOps**— The Rise of DevSecOps

Scott Kennedy

DevSecOps is based on a simple basic principle: everyone is responsible for their own security. It's a mindset that, if adopted and engrained in corporate culture, will result in higher levels of security and faster development. It's also a mindset that requires a rethinking of roles and responsibilities within an organization.

"It used to be that developers developed applications, then security did the security assessment on them. Developers were just responsible for developing the really cool applications," explains Scott C. Kennedy, security scientist at Intuit Information Security. "Now, the security onus is on the owner, the person closest to the cloud account.

"If you think about it honestly, the rush to get your product to market is often the primary driver so security can fall by the wayside. In reality, what is the number one worst thing that could happen with your lovely new product/application/website? A security breach. Let's do something about it then. Let's move from patching broken code, to protecting ourselves as the code is built, tested and delivered."

This is where DevSecOps comes into play. "DevSecOps is a way of using the concepts of DevOps in a security mindset," says Kennedy. "It's often the case in a traditional security environment that the security people are the ones who are the preventers of performance. You have the DevOps team trying to get their work done, but you have teams of security people who say 'no.' With DevSecOps, you create a situation where developers and security people can work at the same velocity.

"Developers can go as fast as they like within the guide rails setup by security. The security team is out in front developing solutions and are the team to follow. If a product team chooses not to follow a policy because it will slow their service down, and if they can get approval from the board of directors and the risk audit committee, and they accept that the trade-off is worth the risk, then security is not going to stop them from doing that. However security will continue to score them as deficient and that will affect their overall scoring. Security is now in the job of informing the teams about the risks of their actions."

> "Let's move from patching broken code, to protecting ourselves as the code is built, tested and delivered."

Intuit has been developing DevSecOps principles for the past few years, coinciding with the company's migration of its core applications to the cloud. "The

entire company was moving to the cloud and we were looking for the best way to do that," Kennedy recalls.

"We found that there just weren't enough security professionals to embed into the DevOps teams, while standard security operations did not work at scale and the continuous change became overwhelming. So we took a step back to ask ourselves what helps us to move the needle? What helps the team do the best they can in the least amount of time?"

A starting point was to put Intuit's development concepts in place and to prioritize its requirements. This meant talking to cloud infrastructure provider Amazon Web Services (AWS) and laying down some needs.

"We went through all the AWS services and reviewed them to see how they met our requirements," explains Kennedy. "We asked, 'Will this work or not work if it has the following features?' Then we started working with AWS on new features.

"As a large enterprise financials company, we had different requirements than some other AWS customers, so we had to talk to AWS about roadmaps and requirements. We couldn't have our data stored in the cloud without specific things in place.

> "Security is now in the job of informing the teams about the risks of their actions."

"We needed CloudTrail logs for everything for example. We wanted to have customer managed keys in KMS. We started pushing Amazon Web Services (AWS) into a much more enterprise-friendly mindset. With CloudTrail, we were able to see how frequently keys are being requested and for what. We could see if people are trying to access too many things."

The firm also implemented a scoring system for development that has implications up and down the hierarchy. "For every one of our more than 900 cloud accounts, we do an active score," says Kennedy. "We can track that last night you stood up 14 instances and we've looked at this and your account gets a 'D' from a security score. That 'D' then rolls up through the division so that it gets a 'D' as well, then up to the VP who also gets the same 'D'.

What this does is to make security everybody's issue, not just the responsibility of security specialists. To that end, Intuit has put in place a Red Team, a small group of people who are charged with testing the firm's security readiness to breaking point.

"We schedule days where the Red Team goes out and attacks the corporation from the outside," says Kennedy. "They go as far and as deep as they can. They can phish executives. They can drop malware. We have a very strong team of people and one day a week we let them loose. Everything is directed by our cloud security team who decide that today we're going to focus on these six people or on this particular service. Everywhere they find a vulnerability, they assign a ticket for it. We are hacking ourselves.

"People don't know whether it's the Red Team or not when an incident occurs. We do constant attacks, we open up tickets and problems go into the development chain as a priority that must get fixed within 24 hours. It doesn't matter if they're in the middle of a big release, this is something that must be fixed now. We have had critical vulnerabilities that have been found and resolved within two hours. We're not producing a 2000 page report for the audit committee here. This is actionable for the development teams."

This has had a positive impact on the overall corporate position on security, he adds. "The success of the Red Team allows the developer community to understand that the entire organization is an organism that works as a whole. Security isn't just something for the people in Building 2," he says.

"Support for such actions and for the adoption of a DevSecOps corporate culture comes from the very top."

"We did a phishing incident recently where the Chief Information Security Officer knew about it and maybe the CEO did. We sent out a very finely-tuned, well-crafted phish and spun up an internal security incident on the back of it. The rest of the security team didn't know about it and HR didn't know about it. That meant that they had to do a fire drill for a serious problem.

"At the end of the day, when we said 'that was us,' we had a small group of people who were upset because they've had to work so hard, but then they learn from that and you have a better rapport. So you hope then that next time any incident will be resolved in minutes, not hours."

Support for such actions and for the adoption of a DevSecOps corporate culture comes from the very top. "The security team reports in to our corporate counsel who has requested that with every single campaign that we run, she is to be target number one. So we attack her laptop first," states Kennedy. "A few months ago we compromised her laptop and replaced her desktop with 'I Heart Red Team.' It was a configuration issue. We managed to take control remotely. Within an hour, access to the entire organization had been disabled. That's a muscle that needs to be exercised regularly to keep it maintained."

"Security inevitably remains an ongoing challenge for organizations such as Intuit," concludes Kennedy, "but the DevSecOps mindset helps to meet that head on."

"One day I hope that the Red Team goes to work and can't find any problems, but they are still finding corners where they didn't think about security and it needs a fix," he says. "In the process, security within this company has gone up dramatically and that allows us to better protect our customers."

Five DevSecOps Organisational Characteristics

1. A customer-focused mindset that tailors security programs and outcomes to customer needs and business outcomes.
2. Scalability—security needs to evolve to keep pace with business outcomes through the introduction of parallel working style and conditions for security teams.
3. Objective criteria that allows business executives to know how, when and in what order to improve the security profile of its business resources.
4. Proactive hunting through the introduction of Red Teams that scopes out an organization's technology environment to prioritize remediation efforts.
5. Continuous detection and response that provides monitoring and analysis of external attempts to attack company targets and allows for incidents to be defeated quickly.

Cargo Cult 101

James Betteley SV-PW

"We have some DevOps engineers who take care of the DevOps stuff."
"We're doing some DevOps things like infrastructure automation."
"We're investing heavily in DevOps tooling!"

Sound familiar? It's quite possible. But if it's any consolation, if you work in an organization where this sort of language is used, or if you've even caught yourself saying these sort of things, then take heart. You are now an honorary member of the DevOps Cargo Cult!

That's the upside. But don't rejoice too much though, because the downside is that as far as cults go, it's not the best one to belong to.

In case you don't know what a Cargo Cult is, here's my Cargo Cult 101. The idea behind a Cargo Cult is the belief that if you mimic and worship some observed activity/events/phenomenon, then of course great rewards will follow. The key point here is the idea that you simply mimic and worship rather than seek to understand how something works. A popular example is of some Pacific Islanders during the Second World War, who, for the first time in their history, came into contact with Japanese and Western cultures when occupying forces used their islands as air bases during the war.

With the arrival of these air bases came a dramatic change in their culture as suddenly an abundance of lavish foreign goods were bestowed upon them. But, as suddenly as they arrived, they equally suddenly disappeared once the war was over.

So what did the islanders do? They simply reconstructed the airbases, control towers and all, out of whatever materials they could lay their hands on (wicker airplanes, coconuts for ear-defenders, that sort of thing) and tried as best as they could to re-enact the activities they had observed in the hope that the supply of lavish foreign goods would return.

Nice idea in theory, but it didn't work in practice, of course.

So what does this have to do with DevOps? Well, let's look to the agile world and its own version of the Cargo Cult. This is where so-called 'agile teams' do all the agile trimmings without actually understanding why.

It all works fine until something starts to go wrong. In this scenario, rather than using a deeper understanding of the principles of agile to allow them to make the right changes and adapt, they instead perform their observed agile activities with even greater rigor, clinging to a belief that the activities themselves will deliver them from their predicament.

This is usually coupled with a belief that the reason why things started going wrong in the first place was because they weren't blindly following the practices of agile with enough rigor in the first place. It's more than a vicious feeding cycle; it's a tough situation from which to break free.

Unfortunately, those of us in the DevOps world are not immune to a bit of Cargo Culting ourselves and as with the "agile Cargo Cult," the problem is rooted in a lack of a deeper understanding of the subject matter. Add to this an unhealthy obsession with observed practices (or "the shiny things" as some call it) and it's a recipe for problems ahead.

I spend a fair amount of time at DevOps meet-ups and conferences and I regret to say that over the last couple of years I've seen an increase in DevOps Cargo Culting. On a personal note, that's a pretty damning reflection on me because it's my job to go around preaching about what DevOps actually is.

To put it simply, I think, to many people, DevOps has become synonymous with automation. To make matters worse, I think this perception/relationship is actively being pushed by various vendors because it's an achievable tangible, easier-to-do and easier-to-sell.

> "To put it simply, I think, to many people, DevOps has become synonymous with automation"

In fact, I was speaking at a DevOps conference a while back and one of the event sponsors carried the tag line "DevOps, automated." It struck me as being a little out-of touch with the general message of the conference itself, which was (unofficially) "DevOps is a culture."

In situations like that, I like to encourage people to substitute the word "DevOps" for the word, "culture." Give it a try and, if what you're hearing sounds like B.S., then just maybe that's because it is. "Culture automated" doesn't sound quite so plausible does it?

Another thing that concerns me is the number of organizations who tell me they're undergoing a "DevOps transformation," but the operations team isn't involved. Which begs the question of what exactly do you mean by DevOps transformation?

In my experience, it's usually something involving one or more of:

- The adoption of continuous integration practices
- Deployment automation
- A move to virtual infrastructure (not necessarily involving cloud)
- Adoption of cloud infrastructure-as-a-service or platform-as-a service
- Some automated infrastructure provisioning

Now, that's all very well and good. These are good things, but they are not DevOps as we understand it. By way of contrast, it's also worth noting the things that we don't hear so much about, such as:

- Monitoring and alerting
- Metrics
- Capturing operational requirements

You could of course argue that these are no more DevOps than the things in the other list, but I do find it interesting that in the DevOps Cargo Cult world, they're so often overlooked.

So why should this be? Is it attributable perhaps to the ongoing lack of agreed definition of what DevOps actually is? Is it a by-product of vendors using their own definitions in order to sell their products and services? Or does the answer lie closer to home with the emergence of a development-centric view of DevOps and a culture based on the pursuit of quick wins? Or is it as simple an explanation as management just not getting it?

Whatever the reason, the question now is what do we need to do to counter the Cargo Cult syndrome? A good starting point is going to be about putting more effort into education and more particularly education of the right people. At this point we need to face up to the fact that education in the DevOps space is lacking to say the least and we need to address this head on. The DevOps community needs to become more enlightened and not look to DevOps certification which might well make the problem even worse. If we look again to the agile world, agile certification has only served to fuel the growth of the Cargo Cult there.

Ultimately we need to ask ourselves, as a community, are we all pulling in the right direction? Until we can truthfully answer yes, then the Cargo Cults will be an unfortunate feature of the DevOps landscape.

"Another thing that concerns me is the number of organizations who tell me they're undergoing a "DevOps transformation," but the operations team isn't involved"

From Continuous Delivery to Business Advantage

Gojko Adzic

Faster isn't always better. In fact, sometimes it's not good at all.

This "faster is better" mentality, when applied to continuous delivery, limits the potential benefits that can be delivered to business, argues software delivery consultant Gojko Adzic, author of several books on impact mapping, specification by example, behavior-driven development, test-driven development and agile testing.

Adzic cites the case in point of a media company where the developers adopted a continuous delivery model, which resulted in them moving from releasing upgrades every few weeks to every few days to several times a day. As a result of this speed increase, the developers assumed they were giving their business users a shiny new toy to play with.

In reality however, the business users found themselves trapped inside something that was moving too fast and too frequently. "You shouldn't confuse users," argues Adzic. "You need to think about what people actually want in terms of continuous delivery. They don't want the ground to move. If it must happen, they want it to happen in a way that they don't notice it is happening."

> "You need to think about what people actually want in terms of continuous delivery"

An update is never going to be as important as the work a user is doing now, notes Adzic. For example, for the content managers at the media company cited, what mattered to them was that they were half way through writing a story and now they had to log out and log in again.

"Continuous delivery has serious side effects that software developers don't always consider," observes Adzic. "Continuous delivery mustn't ever be in conflict with the business. If there's a business reason for a technology refresh, then that's fine but not if it's just for the sake of change.

When you talk to business people about how continuous delivery is going to reduce technical risk, they love it. The trick is to make sure that the human element of control is retained and that users are not left feeling that they are being force-fed changes for their own sake. "When stuff is continuously being pushed at users, then they have lost control," advises Adzic. "You need to think about things outside of the technical pipeline. If we approach continuous delivery purely from a technical perspective, we can mess up really, really badly."

One unfortunate side effect of continuous delivery can be a change in the attitude of users who begin to expect too much for nothing. Adzic points to the real-world example of Tesla when it announced new features for its Model-S range last year. This prompted

> "If we approach continuous delivery purely from a technical perspective, we can mess up really, really badly"

a negative reaction from a customer in the US who had just bought a Model-S a week or so earlier which didn't come with those features.

Feeling this was unfair, he started a petition to force Tesla to retro-fit the new features into the older car. He earned a lot of support from other customers in a similar position and won a lot of attention.

But the point here is that these customers had a sense of entitlement that drivers of most motor vehicles don't exhibit. The owner of a five year old Mercedes is unlikely to demand that the car manufacturer fits all the new features of a later model free of charge onto their old car. But Tesla customers seemingly feel this is their due.

"As we do continuous delivery, users start to expect stuff that's unrealistic," says Adzic. "Whether they're right or wrong, disappointing users and making them angry is not a good thing to do. As you move away from a system of versions releases, users start to feel more entitled. If you have one or two upgrades every year, then people expect to pay and are happy to pay.

"Continuous delivery means that they expect things for free," he adds. "So with mobile apps, for example, you can sell them initially, but then you end up with the development organisation having to deliver upgrades for free. Economically that just doesn't make sense of course."

What needs to happen is a decoupling the traditional ideas of releases and deployment, which contrary to common perception, are not the same thing. If this decoupling does take place, it can have a powerful impact on an organization's marketing.

"For most organizations, deployment and release are the same thing. This is a constraint of old world thinking," says Adzic. "But actually, deployment is the technical event of moving code to production. Release is about the marketing event. It should be done at the point of maximum impact. If you consider they're not the same, then you can design your systems around that.

"You have a continuous delivery pipeline that moves to production, then you can leave the marketing team to decide when to schedule the release. It creates an opportunity for marketing to do gradual releases and validate that something is a good idea. You can become more considerate about how you plan releases. You can organize around marketing events, not technology events. You can ship stuff in production and prove to users that there is no rush."

"This ability to plan releases for maximum impact offers greater brand and profile opportunities," he adds. "When people move away from delivering a couple of times a year to a continuous stream of releases, then it takes the drama away from the releases, but also the excitement. It becomes harder to get news sites to cover your new releases. There can be lots of good stuff in the upgrade from a marketing perspective, but it becomes difficult to get media coverage for small releases."

It's good here to work to a pattern of multi-version development which enables new releases and upgrades to co-exist with older versions, allowing users to proceed at their own pace. This does result in a greater challenge for the DevOps team. "The biggest implication for

DevOps is that you have to think about how to work with multi-versions," explains Adzic. "It's a simple problem to state, but it's difficult to do the architectural changes. You're no longer just expecting one version; you're going to have to support multiple versions."

So DevOps teams need to think in terms of backward compatibility for upgrades to allow users of different releases to work together without issue. But it does allow phased releases of new functionality that provide time for "good ideas" to be put to the test.

This really matters as the case of Google's "41 Shades of Blue" incident in 2009 proves. At that time the firm's head of design wanted to trial a new shade of blue for its links. The developers ended up testing 41 different shades with lots of tension between the developers and the UX team. At stake was millions of dollars of ad revenue.

At the end of day, what emerged was that had the firm gone with the wrong shade of blue and Google was looking at $200 million of damage to its bottom line. As Adzic points out, "When you're thinking about whether something is a good idea, you try it out on focus groups, but you never really know whether people will actually buy it."

"With multi-versioning and continuous delivery and you do not have to publish everything to everybody at the same time," says Adzic. "You can release a feature to one set of users, then to 10 percent, then to 20 percent, and so on, so you can test whether something really is a good idea before you release it.

"If you do multiple versions, that allows the marketing people to decide when something gets released and to who. You consider what happens when you have continuous delivery, what impact it has on marketing and on users. That way you can really unlock the potential benefits to the business."

Threatening Behavior

Fraser Scott

Threat modelling is a structured and methodical approach that identifies potential threats to applications, classifies them by risk and prioritizes mitigation efforts based on the technical and business impact these threats would pose to an organization should they be carried out.

But the traditional approach to threat modelling falls short in a number of key aspects, most notably having the threat identification process take place too early or too late in the development process. On the one hand, it might be carried out during the requirements gathering phase; in other cases, it might not take place until after an application has moved into production.

One reason for this has been the lack of automated tools that can be used to integrate threat modelling into existing workflows. This has, in turn, limited the uptake of the threat modelling concept to a limited number of organisations. Within those, the most common approach is to cultivate proprietary in-house methodologies in the absence of automated tools or industry standards.

There have been some models that have emerged, most notably STRIDE, whose name is derived from six threat categories:

- S is for spoofing identity, such as illegally accessing and then using another user's authentication information, such as username and password.
- T is for tampering with data, including unauthorized changes made to persistent data, such as that held in a database or the alteration of data as it flows between two computers a network.
- R is for repudiation threats, such as users who deny performing an action without other parties having any way to prove otherwise.
- I is for information disclosure threats which involve the exposure of information to individuals who are not supposed to have access to it.
- D is for denial of service attacks.
- E is for elevation of privilege where an unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system.

Approaches to threat modelling that are often used in an adversarial approach are penetration testing (white box and black box) and code review. The immediately obviously flaw in the plan here is that these can only be used once the application has actually been written.

"But the traditional approach to threat modelling falls short in a number of key aspects, most notably having the threat identification process take place too early or too late in the development process"

So if the use of such techniques does throw up security-related issues that need to be addressed, the only solution will be to rework or rewrite the code.

A further complication stems from the fact that such traditional approaches are more suitable to the waterfall development model than anything agile today.

What's needed is a change of approach so that threat modelling mechanisms are actually built into the code as it is being written so as to flag up issue during the development process. This has the beneficial effect of also making the development team more security-conscious.

One solution may lie in the form of code-driven threat modelling, something proposed by Fraser Scott of ThreatSec.org. According to Scott, there is a need to close the gap between development and security by bringing the threat modelling process further into the development process. This, he argues, can be achieved by having developers and security engineers write threat specifications alongside code, then dynamically generating reports and data-flow diagrams from the code.

Security testing is shifting left, from annual pen tests to the realm of unit testing and test-driven development by taking advantage of automation and agile practices. Scott says, "This is about looking at where DevOps and security go now. It's about moving security testing towards the developers and having an automation framework that runs security tests on every single component."

This isn't something that many organizations are doing today. He adds, "It's probably just that not that many businesses have thought about it. People don't threat model when they should. You need to make it part of the established workflow."

"Security testing is shifting left, from annual pen tests to the realm of unit testing and test-driven development"

That is going to require some rethinking of organisational structures and hierarchies. "This starts as a grassroots thing with the techies coming back and giving it a go and saying, 'Let's do this.' It's going to have to be a grassroots concept," says Scott.

"Within the organization the idea is that we should be breaking down the barriers and functions. Developers, QA and security should all be working together, which doesn't mean they need to have six hour meetings. Upper management needs to establish a culture of collaboration.

"It's also a concept that might run in to some resistance from the security establishment. "There is an 'old school' of security who like their little world of responsibilities and have a tendency to say no. To them, this approach might be uncomfortable," suggests Scott. "But security professionals realize that shadow IT is a real risk. You can build all your firewalls but that's not going to help if people by-pass them.

"This idea of code-driven modelling is still experimental," admits Scott, "but it's one that has received some enthusiastic interest from Adam Shostack, author of *Threat Modeling: Designing for Security,* the definitive work on threat modelling to date." http://threatmodelingbook.com

Join the conversation at http://threatspec.org

When LeSS is More

Venkatesh Krishnamurthy

"In this age of agile development, there's an uncomfortable reality that needs to be acknowledged and that is that large scale development is still with us. That's a problem that needs to be faced up to," argues agile coach, Venkatesh Krishnamurthy.

"Modern product development is done through short cycles incrementally building up the product. Small teams are considered better. That said, a lot of people live in a world of large product development. I've not come across a smooth-sailing, large scale project or product development," he states. "They are all plagued with funding, coordination, structural and cultural issues. With so many moving parts, it is difficult to come up with a single formula that could make large-scale program work."

At the lower end of the spectrum, smaller teams are able to turn to the Scrum methodology. For smaller teams, Scrum based on empirical process control has not only proven itself but has become the most popular methodology. But for larger scale programs, something more is needed. That more is LeSS.

LeSS is Large-Scale Scrum, a framework created by Bas Vodde and Craig Larman based on their work in the telecom and finance industries. They worked around the basic idea of needing something that is based on Scrum but which could scale to meet the needs of larger product groups.

Where it differs from Scrum is it adds a more concrete structure with the aim of maintaining transparency while emphasizing the "inspect-adapt" cycle to allow groups to improve continuously their own ways of working.

"From the learning perspective, LeSS provides a clear and concise framework," explains Krishnamurthy. "LeSS is about taking Scrum and asking how we can apply the same concepts across multiple teams?"

LeSS consists of:

- LeSS Principles
- LeSS Rules (defining the LeSS Framework)
- LeSS Guides
- LeSS Experiments.

It also comes in two flavours: LeSS and LeSS HUGE. The former can be applied to up to eight teams, while the latter is appropriate for a couple of thousand people working on a product.

"LeSS also brings good ideas and principles from systems and Lean thinking," says Krishnamurthy. "There are several large scale frameworks out there trying to provide solutions only from a delivery perspective. They won't address the root causes of the problems.

> "With so many moving parts, it is difficult to come up with a single formula that could make large-scale program work"

"LeSS does not promise a quick fix solution, but rather it provides the rules and frameworks to build a strong foundation for the organizations to succeed with large scale development."

And at the centre of LeSS and its evolution is Scrum. "Large-Scale Scrum is Scrum; it is not 'new and improved Scrum,'" insists Khrisnamurthy. "Rather, LeSS requires examining the purpose of single-team Scrum elements and figuring out how to reach the same purpose while staying within the constraints of the standard 'Scrum rules.'

"Many practices from Scrum relevant for single teams are applicable in LeSS with multiple teams," he adds. "For example, single product owner, single product backlog, one potentially shippable product increment at the end of each Sprint, cross-functional teams with the ability to deliver end to end work, stable teams and so on.

"There are minor differences in the way Scrum ceremonies are conducted in the context of LeSS and this is mostly to make it work in the multiple-teams scenario. For example, Daily Scrum is done independently with each team, ceremonies like Sprint planning, review don't need all the team members but key representatives with Product owner."

There are organizational implications that need to be factored in, advises Krisnamurthy. "The structure of the teams, groups and roles they play are critical in driving the team's behavior. That is why, LeSS has a special place for structures. LeSS recommends organizations focus on structures first before rolling out LeSS.

"Some of the LeSS recommended ideas around structure include organizing the team by customer value as well as creating feature teams rather than component teams. A number of deep Lean ideas around management are also evident in LeSS, such as the concept of managers as teachers who create a culture of improvements for teams."

Krishnamurthy is quick to point out that LeSS has not been conjured out of thin air as a response to a problem. Rather, it has evolved across decades of experimentation in different contexts. "I have personally been involved during the initial days of LeSS experimentation, working in a services industry with geographically distributed teams in USA, Germany, France and India," he says.

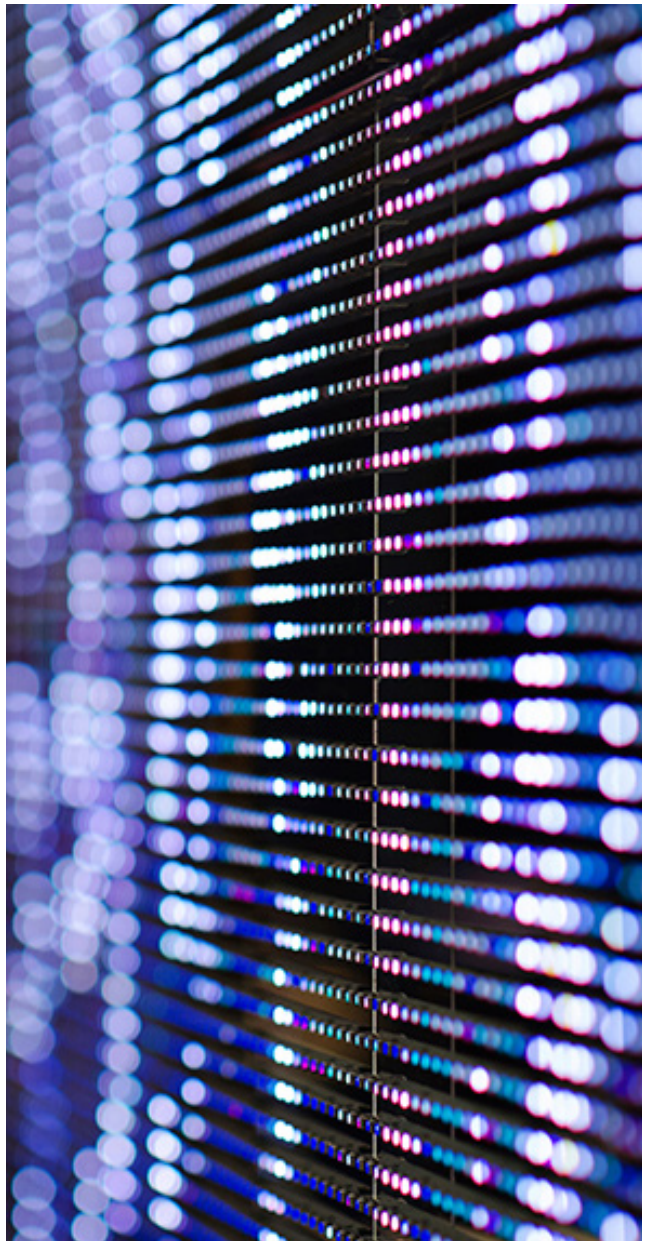
There are several case studies worth looking at from different parts of the world, he adds, including J.P. Morgan's (JPM) Global Core Processing Technology's 3000 plus strong development organization which adopted LeSS in 2013. (See http://www.infoq.com/articles/large-scale-scrum-jomorgan), agricultural machinery firm John Deere (http://less.works/case-studies/john-deere.html) and Telecom Australia (http://less.works/case-studies/telecom-australia.html).

LeSS has been covered extensively in two books, and Krishamurthy highly recommends checking out Less. Works (https://less.works/) and *Scaling Lean and Agile Development* (http://www.amazon.com/Scaling-Lean-Agile-Development-Organizational/dp/0321480961).

Containing DevOps

Matt Saunders

"The role of containers as a game-changer in DevOps can't be doubted to the extent that any provider of software development tools and services needs to find a way to tap into the zeitgeist."



Containers enable developers to work with dev, test, stage and production environments identically. Among the perceived benefits of containers is that they enable more efficient use of server resources and make moving applications easy. Containers also free developers up to work on strategic efforts because they don't need to worry about infrastructure or which cloud provider they are using. But there are also issues that need to be addressed and limitations that need to be recognized so that containerization doesn't become yet another silver bullet fired without target.

One reason for the higher profile of containerization is the rise of the open-source Docker, whose containers run on everything from physical computers to virtual machines, bare-metal servers, OpenStack cloud clusters and public instances. This has some high-level supporters. For example, Amazon last year integrated Docker with its own Elastic Beanstalk Platform as a Service (PaaS) offering. Amazon CTO, Werner Vogels, pitched this as a "best of both worlds" solution for DevOps who "love Docker's impact on their development workflow: packaging applications as Docker Images makes them portable, reliable, easy to share with others and simple to test." (See http://www.allthingsdistributed.com/2014/04/docker-in-elastic-beanstalk.html.)

But it's important not to get too caught up in the idea that containers equals Docker. Other containers, as they say in advertising circles, are available or set to be available soon, such as CoreOS's Rocket, Canonical's OpenStack-integrated LXD containerization engine for Ubuntu and Microsoft®'s Drawbridge container tech.

An interesting insight into the current containers landscape can be found in a recent study, The State of Containers and the Docker Ecosystem 2015, produced by Ruxit and O'Reilly Media. This is hardly a full-scale study polling as it does only 138 organizations, but it does illuminate some indicative trends.

For example, more than 93 percent of respondents are already using or plan to use containers for development, testing or production while within the next six to 12 months, over half (53 percent) of all respondents plan to adopt containers in production.

That's the upbeat news, but of equal interest are the areas that need more attention. For example, some 46 percent of respondents still see challenges around performance management in the production environment. The study also finds that the dynamic nature of containerized environments increases the need for more reliable and production-ready solutions for orchestration (56 percent), performance monitoring (46 percent) and automation (40 percent).

It's also the case that mainstream enterprise applications from the likes of Oracle and SAP won't be moving to containers any time soon. That said, Google is running its cloud using Linux® containerization. At the recent Gluecon conference, the firm revealed that it currently starts over two billion containers every week.

"Containers are a hot topic in contemporary IT, but it is important to understand the benefits and drawbacks of using them," says Matt Saunders of Contino. "Similarly, containerization cannot exist as a sole strategic move, taking the full benefits also requires understanding of microservices, service discovery, environments and resilience."

At a time of budgetary pressure, containerization offers clear benefits. "Containerization is a great opportunity to make better use of scarce resources," notes Saunders. "Perhaps you have jobs that run on a schedule, maybe once a week or once a month. Perhaps you have dedicated hardware sitting around idle, burning money.

"Containers are a hot topic in contemporary IT, but it is important to understand the benefits and drawbacks of using them"

"As containers have all runtime dependencies bundled together, these containers can be started reliably with minimal effort and delay. Innovation in scheduling systems with new tools such as Mesos and Kubernetes heavily leverage the container concept."

He adds, "Putting an application in a container makes it much easier for them to be portable and can run anywhere. Containerizing an application that runs in multiple environments (e.g. through CI, dev, test, staging and production as part of the SDLC) reduces the friction required to in promoting the application towards live."

From a microservices perspective, containerization works well in combination with microservices applications built to the twelve-factor principles (http://12factor.net/). "As containers are designed to be short-lived, an application performing stateless actions (complying with factor six) would fit well into a container," explains Saunders. "Applications that require state retention, or shared state between applications, will be harder to containerize."

An application that can scale-out by adding more servers fulfilling the same role can be simplified by containerization. "Service discovery is an area in which much innovation is currently taking place," says Saunders. "The options for seamlessly sending traffic to scaled-up infrastructure are growing. Centralized applications which can't be scaled in this way have limited benefits from being containerized."

Game changer or another roll of the DevOps dice? It's too early to tell. But containerization is clearly a 2016 DevOps agenda item for further exploration, experimentation and discussion.

Why NoOps Actually Means No Opportunities

Pete Waterhouse—CA Technologies

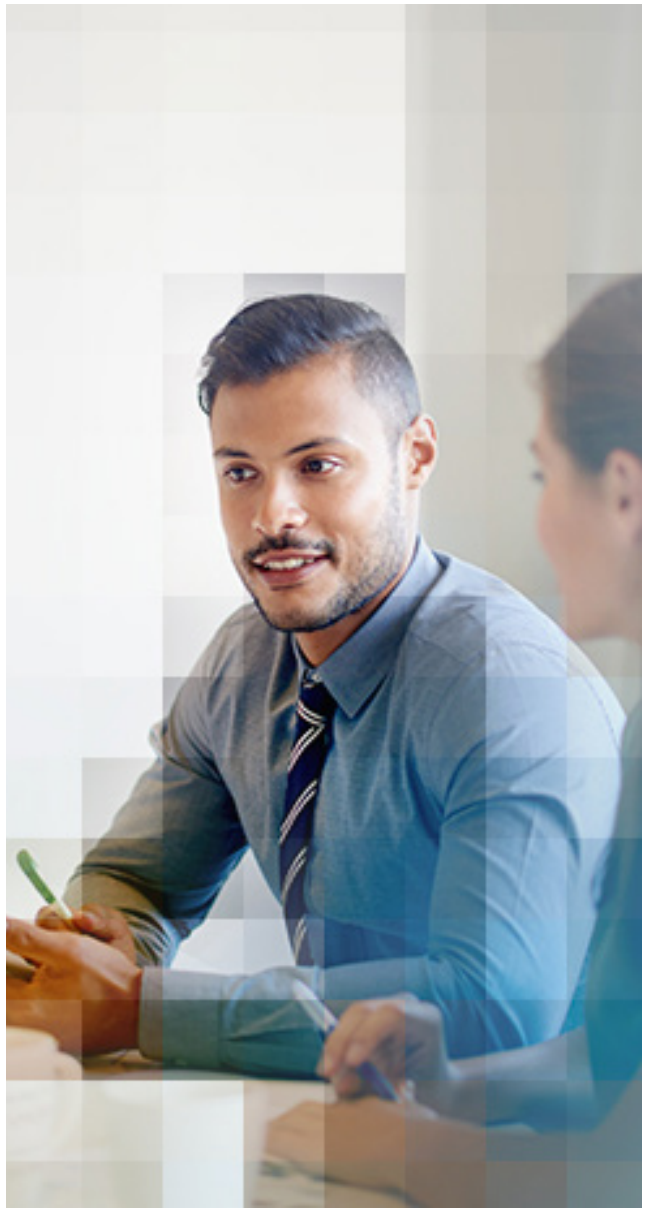
Let's spare a collective thought for all of us who work in this crazy IT industry. Ever since the first hunks of Big Iron hit the commercial sector, we've been struggling (and generally failing) to keep up with the demand for software applications.

Now it's getting harder. Not only do we have to satisfy an insatiable appetite for new apps and services, but the nature of the business palate is changing. Business now demands a continuous flow of software innovation over back-office application support—or, using the haute cuisine analogy—they want IT to help serve a top-end degustation menu versus just warming up a ready-made TV dinner.

Thankfully, the tricks we've learnt in software development over the last fifteen years or so have better equipped many organizations to solve "wicked problems." That is, finding digital solutions to problems that haven't yet been fully defined. And, while that sounds counterintuitive, it's an approach completely aligned to the dynamics of new disruptive business models and digitally transforming government services to improve citizen engagement.

All this bodes well for the adoption of agile as the dominant software development methodology. Wicked problems demand fast iterative style development with fully autonomous teams empowered to increase the rate of software deployments.

Of course, part of this autonomy comes from the philosophy outlined in the original agile manifesto, but technology advances have also helped fuel the fire. Development pros now have open source and cloud platforms to thank for helping increase the flow of value to the business.



With continuous integration, software builds are streamlined, while automated test-driven development has made the notion of parallel development a reality. Add the ability to codify environments (ala infrastructure-as-code) and it appears development has everything needed to build, deliver and run applications.

Sound great—so where does the role of IT operations fit into all this?

Some on the development front (especially those from "Unicorn" organizations) may argue that IT Ops is a completely defunct or redundant function. Even credible analysts and thought leaders have touted NoOps (no IT operations) as a viable option. Some even suggesting that it accelerates DevOps benefits because collaboration is a given when (by removing the Ops function itself) you remove any and all organizational friction.

"Some on the development front (especially those from "Unicorn" organizations) may argue that IT Ops is a completely defunct or redundant function"

Certainly, and in looking at how IT operations is currently positioned and leveraged by many organizations, there is some validity to this argument. After all, if developers can continually infuse operational functions into their coding, why do we need IT operations at all? Perhaps this logic explains why for some, IT operations as a function is becoming disenfranchised; no longer having a voice into critical digital transformation initiatives—or if it does, just a traditional, risk-averse, glass half empty mindset—the last deployment bottleneck.

This logic, however, is flawed due of a couple of inconvenient truths.

Firstly, however operationally awesome developers think they are, building resilience, maintainability and supportability into applications is not always top-of-mind. Worse still, these elements might be completely neglected if the parent organization is fixated on measuring (even compensating) development

"However operationally awesome developers think they are, building resilience, maintainability and supportability into applications is not always top-of-mind."

staff based on software app "speeds and feeds." Secondly, even if these elements are addressed in development, they're often only conducted at the end of development cycles or bolted on after problems are discovered—that's sort of like baking a cake, forgetting the sugar and then trying to compensate with a sickly chocolate sauce.

In reality, great operations engineers are best equipped to help development incorporate operational excellence into their practices. After all they've built expertise gathered over many years supporting every new wave of technology—from mainframes to microservices. What must change, however, is how this expertise develops and is shared across the organization. Here's a few important considerations.

- **Become Cloud Connoisseurs**—Like our development colleagues operations' emphasis will shift from on-premises to cloud. This means ensuring Ops expertise is fully leveraged to support the business nirvana of delivering high-quality digital customer engagement at scale. So if you're an IT operations professional still configuring QoS policies on routers or manually provisioning development environments, it's probably time to skill-up in PaaS, Amazon EC2 and containers. Either that or go and find another gig.
- **Embed Ops in Dev**—This means being less separatist and siloed and more inclusive and unified. Rather than focus on technical diagnostics and reacting to failures, the new operations professional will apply systems thinking to holistically look at how business applications and the all-important customer experience can be improved. This is easier said than done with skeptical development teams, so it's essential that monitoring feedback is automatically shared and incorporated into all practices—for example into agile sprints
- **Embrace Lean thinking**—This involves being less interrupt-driven, fixing one technical problem after another, to working side-by-side with other teams to ensure a constant flow of value is delivered to business with all waste removed. As with our development colleagues, elements of agile (and common sense) thinking will come to the fore—with terms like "never done" and "continuous improvement" becoming established elements of the operations and performance management mantra.

Perhaps the biggest change IT operations will undergo is how it interfaces to other teams. Rather than engaging in low-value and redundant activities, operations will morph into providing sets of easily accessible and repeatable processes other teams will use to bake quality into everything developed and tested. To this end, operations will transform from an insular function designed around "keeping the technical lights on" to a high-value craft that outfits the software factory and staff with new capabilities they need to ensure speed and quality at scale.

For IT operations as discipline, this means letting go of many traditional admin tasks—not just running things, but participating in crafting automated processes their colleagues can use. So rather than routinely monitoring applications in production, the Ops in DevOps will ensure this capability is available in pre-production. Rather than mundanely provisioning individual servers from change requests, new teams will equip the organizations with a complete release environments that support programme level goals.

Operating as DevOps "craftspeople" makes complete sense because it moves expertise out from behind the production curtain and incorporates it into software development and testing. The organizational focus of IT operations positively changes too—from being good at describing and fixing technical problems to being awesome at prescribing improvements that drive better business outcomes.

"Operations will morph into providing sets of easily accessible and repeatable processes other teams will use to bake quality into everything developed and tested."

This strategy also involves re-engineering (or retrofitting) the software factory with capabilities that enable infrastructure and applications to recover from failure and then get stronger. And while this design-for-failure thinking is often considered the domain of pure-play cloud companies and start-ups, it's exactly this kind of operational craftsmanship that'll differentiate the highly-scalable business successes from software also-rans.

Great organizations exploit their business models with flawless operational execution. With these models now being constantly re-shaped by software applications, IT operational excellence is more important than ever and must be crafted into applications at every opportunity.

Relegating the Ops in DevOps with NoOps thinking is a recipe for disaster.

Lindsay Holmwood

Engineering Manager

Lindsay Holmwood is an engineering manager living in the Australian Blue Mountains. Lindsay works at the Australian Government's Digital Transformation Office, helping teams successfully build and operationalise digital services that solve problems for all Australians in a simpler, clearer, faster and more humane way. Convening the second ever DevOps conference in the world in Sydney in 2010, Lindsay is currently the Secretary of DevOps Australia, and has organised the monthly Sydney DevOps meetup since 2010—the longest running DevOps meetup in the world. Lindsay speaks internationally about both the cultural and technical side of DevOps, covering just culture, complexity, cognitive biases and monitoring tools. He also won third place at the 1996 Sydney Royal Easter Show LEGO Building Competition.

Scott Kennedy

Chief Security Scientist

Scott Kennedy is a chief security scientist, cloud security team member, Intuit Cloud Security Team member and an OpenInferno Hacker CTF Game toolkit team member. His certifications include: CISSP-ISSAP, CISA, GCIH and GIAC. He has twenty-five years of experience as a UNIX® system administrator, including 10 years of dedicated security experience within commercial enterprises, SME in fields including, cloud security, virtualization, IT security, document retention, disaster recovery, IA, CNA/CND and forensics. As part of the Intuit Cloud Security Team, he is responsible for bringing scientific rigor and analysis to the decisions for cloud security. He is an active member of the Southern California IT Security scene.

James Betteley

DevOps Evangelist

James Betteley is the author of some of the worst Ruby code ever committed to GitHub (and he's not much use as a sysadmin either). Luckily for the software development world, James doesn't work as a developer or a sysadmin (phew). Instead James makes a living as a DevOps evangelist, trainer and agile coach with Devopsguys, which is thankfully something he's much better at. James has spent the last few years neck-deep in the world of software automation, helping a wide range of organisations to improve their delivery processes through the use of best practices and best-of-breed tooling. He can often be found speaking at various conferences on DevOps, continuous delivery and agile development, and he also authored the chapter "Scrum for Ops Teams" for the book Build Quality In. James often wears a hat.

Gojko Adzic

Strategic Software Delivery Consultant

Gojko Adzic is a strategic software delivery consultant who works with ambitious teams to improve the quality of their software products and processes. He specialises in agile and lean quality improvement, in particular agile testing, specification by example and behaviour driven development. Gojko's book, Specification by Example, was awarded the number two spot on the top 100 agile books for 2012 and won the Jolt Award for the best book of 2012. In 2011, he was voted by peers as the most influential agile testing professional, and his blog won the UK agile award for the best online publication in 2010. Gojko is the author of Fifty Quick Ideas to Improve your Tests, Fifty Quick Ideas to Improve your User Stories, Impact Mapping, Specification by Example, Bridging the Communication Gap, and Test Driven .NET Development with FitNesse.

Fraser Scott

Sysadmin-that-codes

Fraser Scott is a sysadmin-that-codes with a background in systems administration, automation, monitoring and operational security. He has over 10 years of experience in start-up, SME and enterprise environments, working in a variety of industries including telecoms, new media and retail.

Venkatesh Krishnamurthy

Computer Scientist, Agile Coach, LeSS Practitioner

Venkatesh Krishnamurthy is a computer scientist, agile coach and LeSS practitioner with more than 20 years of experience working with several start-ups and IT companies across U.S., Europe, Asia and Australia. Venkatesh has been practicing agile since 2001. He started his journey with XP and, over a period, embraced Scrum, Lean and Kanban. He has successfully applied various ideas from systems thinking and complexity science while bringing changes to teams and organizations. Venkatesh has successfully implemented agile on collocated and geographically distributed teams with team sizes ranging from five to 200 with the ease. He has accumulated rich expertise while applying several scaling strategies across financial, telecom, aerospace, travel and retail industries. He is rated as one of the Top 200 Agile Bloggers on all matters of agility, innovation and complexity. He contributes as an agile advisor and curator for Cutter Consortium, Zephyr, Techwell and DZone. He is also an author and speaks regularly at various agile, Lean and systems thinking conferences.

Matt Saunders

Senior Consultant

Matt Saunders is a senior consultant and heads up the Docker and Containerisation practice at Contino (www.contino.co.uk). With a background in both enterprise and start-up worlds, Matt aims to bring the best of DevOps and continuous delivery ideas to software delivery teams of any size. Matt is also co-organiser of the London DevOps meetup—a group with over 2,000 members which meets monthly.

Pete Waterhouse

Senior Director, DevOps Strategy and Marketing

Pete Waterhouse has been involved in the development, support and marketing of software solutions for more than 20 years. He has held a number of management, consulting, technical sales and strategy positions in areas such as cloud computing, DevOps and IT business management. Pete blogs on a range of disruptive business and technology trends, with articles appearing Information Week, Wired Insights, DevOps.com and App Developer Magazine.

Next Steps

Mainstream adoption of DevOps is here. Is your organization ready to seize all the business benefits and opportunities it presents? At CA Technologies, we have built a portfolio of products and solutions on our DevOps expertise.

Visit ca.com/contact to learn more about how CA can help you close the gap between your developers and your operations—and keep your competitive edge in the application economy.

CA Technologies (NASDAQ: CA) creates software that fuels transformation for companies and enables them to seize the opportunities of the application economy. Software is at the heart of every business, in every industry. From planning to development to management and security, CA is working with companies worldwide to change the way we live, transact and communicate—across mobile, private and public cloud, distributed and mainframe environments.

For more information on DevOps solutions from CA Technologies, go to: ca.com/insights/devops

#BusinessReWrittenBySoftware

CS200-163317_1215