ca technologies

DevOps Perspectives 3

Straight talking and the latest thinking from the DevOps frontline

CONTENTS

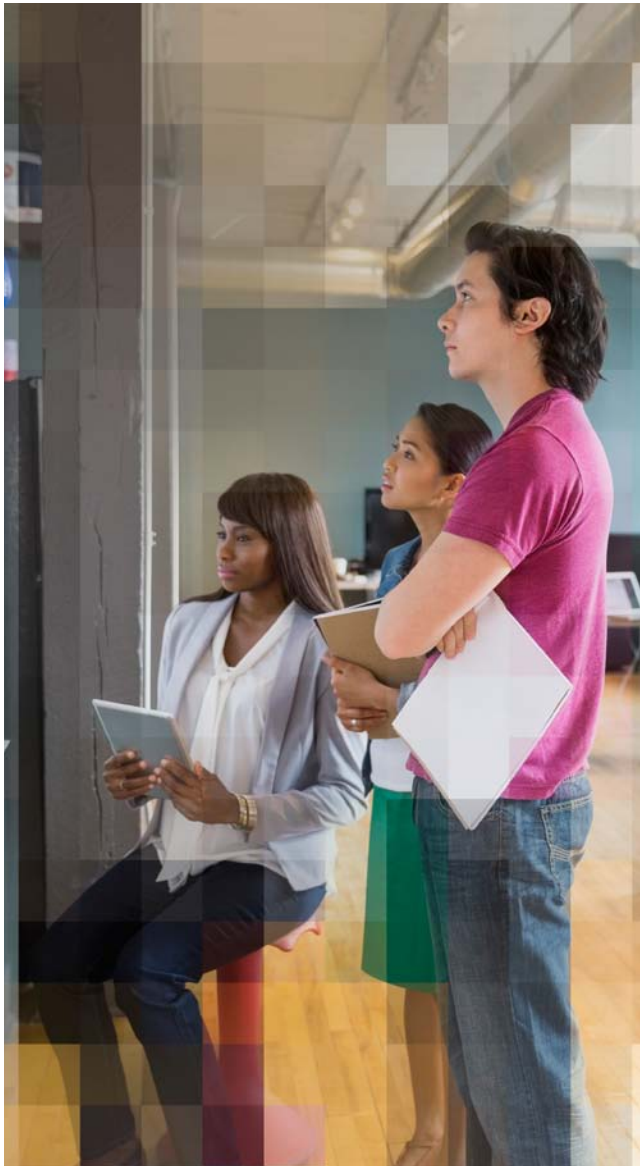
DevOps – mainstream tipping point ahead

There's some evidence that DevOps as a core part of mainstream IT strategies is not that far away, although as Em Campbell-Pretty points out on page 22, there's still a lot of 'de-training' to be done with the CIO first!

Recent articles predicting the increased enterprise adoption of DevOps and the formation of the DevOps Institute point to a maturation of DevOps as an approach. One key requirement for adoption is incentivization and how employees are objectivized on success and how this affects their behaviors, which James Smith of DevOps Guys explores on page 19. Equally, the creation of the DevOps Institute has created some stir in the market, with views for and against the need for some form of certification of capabilities in this area being expressed widely. Read more about this debate on page 12. Microservices as a hot topic has come to the forefront in the last 12 months or so and it's one we have not just one but two expert opinions on, with Matthew Skelton and Jason Bloomberg providing the consultant and analyst views on pages 10 and 16 respectively.

One popular definition of DevOps uses the acronym 'CALMS', standing for the topics of Collaboration, Automation, Lean, Measurement and Sharing. The 'Automation' subject is one which has seen a great level of buy-in with various vendors offering automated capabilities in the build, continuous integration, environment provisioning and release management areas for example (CA Technologies being one of these of course). Dan North explores where and how automation is a good thing, and also advises where to adopt a more cautionary approach in his excellent article on page 25. Lastly, let's not forget that the DevOps philosophy hinges on people, process, technology and information, an argument also made by Nicole Forsgren on page 7.

We hope you find this issue as stimulating and thought-provoking as previous editions.



DevOps – the recruitment challenge

Grant Smith, Author, Next Gen DevOps
Tony Chapman, Managing Director, LinuxRecruit
Patrick Hyland, Founder, DevOps Associates

It has been argued that DevOps is in danger of becoming too 'elitist'. Some have suggested that DevOps people are entitled to be treated differently. But is this a view that finds favor across industry?

And is the 'elitist' argument masking a more concerning issue — that of organizations pursuing the elite DevOps title at the expense of missing out on highly experienced candidates who are rejected due to not having the right job title?

Grant Smith, author of Next Gen DevOps, can understand how the elitist charge against DevOps has come about. "When a less powerful group challenges the perceptions and privileges of a more powerful group, there are always accusations and misinterpretation," he says.

"DevOps demands that development and test teams become involved in the management of services throughout their lifespan. Those CTOs coming to their role from a product development or management background struggle with this concept because they don't understand the difference between an application and a service."

> "Is being elitist necessarily always a negative thing?"

Tony Chapman, Managing Director at LinuxRecruit, poses an intriguing question when he asks: "Is being elitist necessarily always a negative thing?

"Any new movement at the cutting edge needs leaders to define the approach and disciples to follow. DevOps needs the Elite to master the approach and teach others; as long as those at the pinnacle allow others to follow, it should be healthy for the DevOps community."

Patrick Hyland, Founder of DevOps Associates, talks less about elitism and more about dedication above and

beyond the norm. "To be successful in any discipline, to go beyond mere mediocrity, one needs to be committed to, passionate and very serious about the work," he argues.

"The Myers Briggs MBTI personality theory may be useful in understanding some of allegations of elitism, both from the point of view of an individual who interprets an interaction as elitist and from the point of view of the person who is being viewed as elitist," he adds.

"There are obvious, maybe objective cases where a person comes across as arrogant and condescending but I think outside of those obvious cases there is a much more nuanced personality-specific interpretation happening," he concludes. "The best of these individuals are tolerant, have developed a mature self-awareness and know how to approach the work with humility and a sense of balance."

"As with anything, if demand exceeds supply, prices rise."

They're also well paid. Chapman sees the higher salaries on offer to those with DevOps job titles as a simple case of market economic realities. "The daily rate debate within any technical industry is a simple case of supply and demand," he argues. "As with anything, if demand exceeds supply, prices rise.

"There is a huge demand for an intricate set of skills within the DevOps toolset, and a short supply of highly qualified individuals. This invariably increases rates. Due to high rates, the contract DevOps market is beginning to be flooded with contractors adding 'DevOps' into their job titles, or other buzzwords in technologies: Docker, Puppet, Ansible etc."

This is hardly surprising, he adds: "If someone is willing to pay people an inflated rate, purely by adding DevOps into their title, you can understand it to a certain degree. The issue is uneducated recruitment consultants, or even uneducated hiring organizations, not understanding what is required to be a true DevOps individual."

Mature DevOps practitioners are rare but the demand is high, agrees Hyland, adding that there's also a good

deal of confusion around what DevOps really means in practice when it comes to recruiting staff.

"There is still too much focus put on technology skills and too little put on mature system thinkers with a technical background," he says. "I think if clients and recruiters phrased the job spec in a different way they may get more of the types of candidates that they and the clients' people want to work with."

This is a point picked up by Smith, who makes the case that DevOps is not a monoculture and, given that people move into the space from different places, there's not an accepted single perception of what DevOps actually is.

"The IT industry as a whole is confused about what DevOps is and what the advantages are and that goes double for CTOs, most of whom started their careers in development or product," he says. "DevOps is challenging the status quo. It is focusing people's attention on the service as a whole, not just on application development and launch."

Smith flags up shortcomings in how recruiters source DevOps candidates. "Recruiters have optimized their sourcing processes so that their tools or their low-paid sourcers can scan through thousands of CVs and LinkedIn profiles, pattern matching for computer science, Java, Git, Jenkins etc," he explains.

"The recruiter can then contact the developer with an almost cookie-cutter job description and the developer will understand what they'll be expected to do. There is no DevOps or Operations degree. There aren't standardized languages or tools yet and so recruiters have a harder job identifying candidates. That means that recruiters are bombarding anyone with DevOps on their CV for every role they have."

> "Being elitist can potentially render some positions literally unfillable."

But there's a lot of DevOps opportunism out there that can pose a problem, argues Chapman. "Being elitist as an organization, without actually understanding what this means, can potentially render some positions literally unfillable," he warns.

"There are companies we have partnered with, who are not as educated or advanced enough in the DevOps arena but who engage us requiring a DevOps engineer. On further qualification, it's clear they just require a system administrator (or another skill set).

"But the system administrators are discounted due to not having 'DevOps' written on their CV. The actual DevOps guys are not interested due to on further inspection it not actually being a true DevOps role. It continues in a cycle, until system administrators need to include 'DevOps' on their CV to be considered, which they are then potentially vindicated for."

This could have long-term negative implications that stem from the pursuit of this elitist status. "There is the possibility of some organizations missing out on great people, purely due to them not having the latest buzz technology listed as a skill," suggests Chapman.

"I see a huge amount of organizations rejecting very good candidates for a permanent position because, for instance, they don't have Puppet experience. For a talented engineer, learning a new technology such as Puppet configuration management shouldn't be the hardest thing in the world, as long as they have a solid engineering background and the right mindset.

"Many of the talented Linux sysadmins I have previously worked with and placed into great organizations are now among the elite DevOps Engineers and visionaries. At some point in the last few years they have learnt how to master CI/CD, Automated Deployments and Configuration Management tools, so arguably we need to give others this opportunity without immediately discounting them."

He concludes:

"Many companies are creating dangerous situations where they are rejecting potentially great people, who could provide a significant long-term ROI, simply due to them not having the latest buzzword on their CV."

DevOps and the bottom line

Nicole Forsgren, Director of Organizational Performance and Analytics, Chef

DevOps is not just about making an impact on the IT function, but also on the bottom line—and that's revolutionary. That's the hypothesis proposed by Nicole Forsgren PhD, Assistant Professor at Utah State University and Director of Organizational Performance and Analytics at Chef, and she's got the data to back it up.

This, she argues, flies in the face of decades of research. "For the first time in several years, research shows a link between IT investment and organizational performance, but only if those investments exist with the right mix of IT, culture and practice, aka DevOps," she says.

"What's interesting is that for decades we've tried to find the value that technology can bring to a business and haven't been able to find it. Investment in IT doesn't impact the bottom line, any kind at all. We just don't see it. Studies fail to show a link, time and time again."

This is a manifestation of what Forsgren calls the Productivity Paradox. "Anyone can go out and buy a server and throw it in the closet," she explains. "But your competitors can do that as well. It's a low barrier to entry, so they can buy the same server or a similar server and so you don't get any real competitive differentiator.

> "You need to have a culture in place that is open and generative and communicative."

"If you do get any kind of competitive advantage, it's just not sustainable because technology advances. So technology never becomes a differentiator for you. What you need to do is find a way to leverage technology—or indeed anything you have—in such a way that you can really set yourself apart from your competitors. The ROI rarely pans out and if it does, you're looking at three, four or five years. It just doesn't work."

But DevOps is different, suggests Forsgren, stating that: "DevOps is not just a technology solution. It's not just a server, it's a major re-engineering shift. DevOps has ended up being a significant process change. Yes, it requires IT investment, but it also requires investment in culture and the re-engineering of processes.

"For change to happen, you have to include the right people and processes, the right culture, the right tools, the right technology. You need to have a culture in place that is open and generative and communicative. You need to have Dev and Ops talking. You have to re-engineer the entire process and you have to do it very mindfully. We don't talk about best practice, but you need to have good practices.

"It's like the lean and the Toyota way of manufacturing that we saw in the 1990s. Just as that revolutionized the way manufacturing was done, DevOps will revolutionize the way IT is done across all industries."

Riding the unicorns and horses

What's also remarkable about DevOps is that its potential reaches across both the 'unicorn' and 'horse' categories of company. "Unicorns are young and nimble. They are start-ups or they were start-ups. They're companies like NetFlix and the way that they do things is the DevOps way," explains Forsgren.

"Horses are major established companies which have been around for so long that they are set in their ways. It's like old dogs and new tricks. You have firms that just can't drastically change the way they do things or manage their IT processes—or so people believed."

Falling under the 'unicorn' banner would be a firm like accounting software provider Intuit, which used DevOps to experiment with new functionality. What was particularly bold was that this experimentation took place during the tax season, the busiest period for the company.

Again, this runs counter to accepted wisdom, but in reality what time is better to do experiments with functionality than the period during which customers are using your products most? In this case, following such a course of action had a business impact of delivering a 50 percent increase in conversion rates on the website.

Meanwhile at a 'horse' company, DevOps has resulted in the ability to deploy code quicker and deliver services to their customers much faster. "They can run experiments in real time so that they can understand what features are more valuable and decrease customer churn," says Forsgren. "They're deploying code hundreds or thousands of times a year rather than a couple of times."

More than just **IT**

DevOps isn't just IT, it's the practice of IT. High-performing IT organizations are twice as likely to exceed the business's profit, productivity and market share goals, notes Forsgren. "You see a change in the business view of IT. It's seen as a cost center at first, where you have to do IT just to keep up, but then it starts to be seen as a point of distinction that can deliver genuine value to customers. You can attract new customers and retain existing ones."

Forsgren observes that DevOps has really been a ground-up, rather than centrally driven, movement. What is now known as DevOps has existed inside companies that have been the best IT performing companies, but just not called DevOps.

"Once it had a name, it became a thing," says Forsgren. "But it wasn't something that came approved from the front page of the Harvard Business Review. It came about from people going to conferences and talking to one another about what they're doing. It's one of the interesting things about the movement: those involved in DevOps talk to each other, and they help each other. They call one another up on the phone and chat, or they go to meet-ups. They leverage one another. They make it happen and then they go public with what they've been able to accomplish.

"DevOps creeps into enterprises. You might not be able to use DevOps across the whole organization all at once, but it gets rolled out a piece at a time. Let's try it here, let's try it there, strategically. You might identify certain applications that are legacy and are always going to say legacy from a business point of view. But there are other bits of the business that can become points of distinction, so those can use DevOps practices and principles to deliver value to the business."

But DevOps is good for the IT function. According to data, high-performing DevOps teams are more agile, with reports of 30x increase in deployments and 8000x faster lead times than peers. They are also more reliable with 2x the change success rate and 12x faster mean time to recovery.

So are CIOs already highly excited by the promise of DevOps then? Not so much, according to Forsgren. "CIOs still need to be educated when it comes to DevOps," she suggests. "A lot of CIOs are still running IT as simply cost centers. They just aren't excited by DevOps and they just don't really care.

"It is changing though. I've had a few people ask me to write up a quick blurb on evidence that it can help with the contribution to the bottom line. Before it used to be 'why bother, why care?' So there is progress."

> "DevOps isn't just IT, it's the practice of IT."

Looking ahead, Forsgren reckons that effective use of DevOps will become a critical business differentiator. "I can see DevOps being part of the strategy for the enterprise," she says. "The smart way to do this is piece by piece. The best way to use it is to take a strategic application and deliver value all the way through the value chain."

But she concludes that there will be enterprises that will choose not to use DevOps in any way. "Some companies will survive because they are so big," she says. "Other companies that resist the DevOps change just won't be around. Right now, adopting DevOps is a point of distinction. For some companies it will be a point of parity. For others, they will just fail. Survival isn't mandatory."

Microservices and data **consistency**

Matthew Skelton, Co-Founder and Principal Consultant, Skelton Thatcher Consulting

Microservices have rightly become a good pattern to adopt when deployability and rapid changes to independent services are important for a software system. However, without a suitable data strategy in place, organizations building microservices risk data duplication and inconsistency where coordination between service teams is limited.

Optimizing for deployability

For software systems that need to change rapidly and often, the microservices pattern is an emerging good practice, particularly when combined with container technologies such as Docker or Rocket. Not only does *deployability* increase, but also Dev team engagement, together leading to more maintainable and evolvable systems. At Skelton Thatcher Consulting, we have been working since mid-2014 with teams at several different clients that are moving towards smaller deployment units (whether microservices or simply smaller, less coupled services) and we have seen first-hand the benefits for teams. The reduction in change complexity is a particularly useful outcome, along with a reduction in accidental coupling.

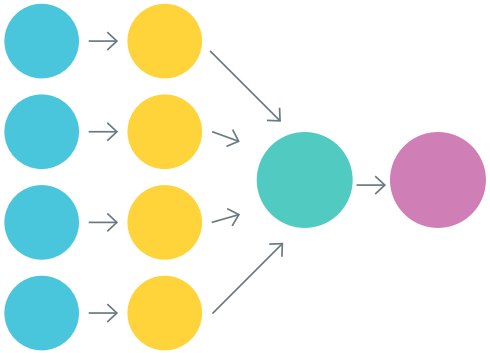
The hidden **value** of a monolith

Unfortunately, in the rush towards microservices, some teams seem to be unaware of the value previously provided by their older monolithic architecture, particularly if a central relational database was used. In many organizations with multiple product or budget streams, it is (sadly) fairly common for the different product streams—or even multiple 'projects' within a single stream—to be effectively in competition with one another.

Now, with a monolithic architecture and especially with a central relational database, these conflicts are resolved at compile time, data load time, or integration test time: at any rate, almost certainly *before* the software reaches Production. Feature requests that conflict (perhaps on a database column name) are batted back to the Product Owners to resolve; in effect, the monolith and central relational database are acting as an arbitrator of feature requests (figure 1).

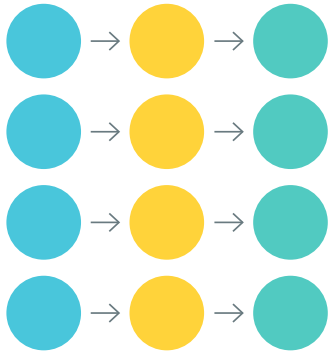
However, in a microservices world, the coordinating effect of the monolith or central relational database are typically lost, potentially allowing changes that conflict at a fundamental business level to reach Production before the conflicts are detected. We have seen on several occasions that product and program managers can be vehemently opposed to deliberate coordination with other product or program managers, preferring to pretend that their requirements are entirely independent of those of their peers (figure 2).

Figure 1



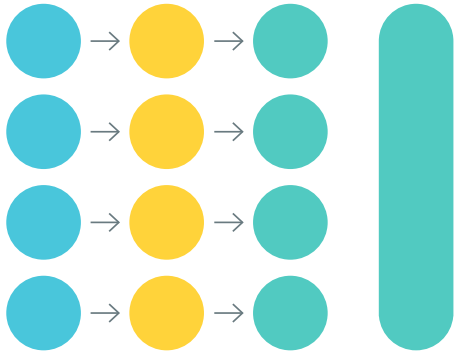
traditional team of Dev, QA, DBA, Ops

Figure 2



microservices teams with data capability within each team

Figure 3



microservices teams with a pan-organization data strategy

Prediction: data strategy for microservices

At the time of writing (June 2015) we have yet to see major data consistency problems emerge from the (mis) use of microservices, although people like Simon Brown are warning that the use of microservices is not a way to avoid good software design. We are likely too early in the adoption of microservices for the problems of data consistency to have emerged for many teams. However, unless different product teams are using some kind of shared bus or event store for cross-service consistency, we predict that the lack of a monolith or central database could result in teams duplicating data and logical entities across different data silos.

Naturally, with a relational database, we can rely on foreign key (FK) relationships and constraints in order to enforce data consistency, but we have much softer constraints in place when working with microservices and split data storage.

This means that the organization can no longer rely on a central relational database for maintaining data consistency but must find alternative methods. Perhaps the role of the 'data architect' evolves to cover pan-organizational data consistency, and maybe we'll see some new tools emerging over the next few years to deal with the difficulty of correlating data from the data silos of different microservices (figure 3).

The issue of pan-organizational data consistency and data integrity is something that cannot be ignored if business outcomes are to be met and sustained over the coming years, even as organizations adopt patterns such as microservices to help achieve more rapid and frequent software changes.

DevOps institutional thinking

Justin Vaughan-Brown, Global Digital Transformation Lead, CA Technologies

In March 2015, I hosted one of our regular 'DevOps Dinners' in London and one of the questions I posed to the assembled audience of customers, partners and thought leaders was whether the newly launched DevOps Institute was a necessary addition to the DevOps movement.

The Institute's mission statement says that it will work with thought leaders from the DevOps and IT Service Management communities, as well as the IT training market to become the standard in quality, enterprise-grade DevOps education.

The DOI founders acknowledge that they are causing a stir. On its website, the Institute concedes: "We recognize that certified DevOps training is new to the DevOps world. But the time has come for DevOps to take its next step."

"To succeed, best practices need to be codified and taught in the time-honoured methods used by IT for over 40 years. Not that DevOps isn't different, it most certainly represents a new way of thinking, doing and acting within IT and organizations as a whole. But that doesn't mean traditional education and training, including certifications, will not be applicable."

Heated **debate**

That's the kind of bold statement that was always going to spark heated debate in some quarters. Certainly not everyone sees this as a good thing. Sam Newman at ThoughtWorks takes particular exception to the emphasis on certification in the Institute's statements of intent.

In a blog posting he writes: "I certainly have no problem with people making money from DevOps in general or DevOps training in particular. If the group had come out and said 'we're going to offer awesome training courses—look at our fantastic content!' I would probably have been very supportive, and may even have pointed people in their direction. However, the Institute sees certification as essential, and it is front and centre in their marketing."

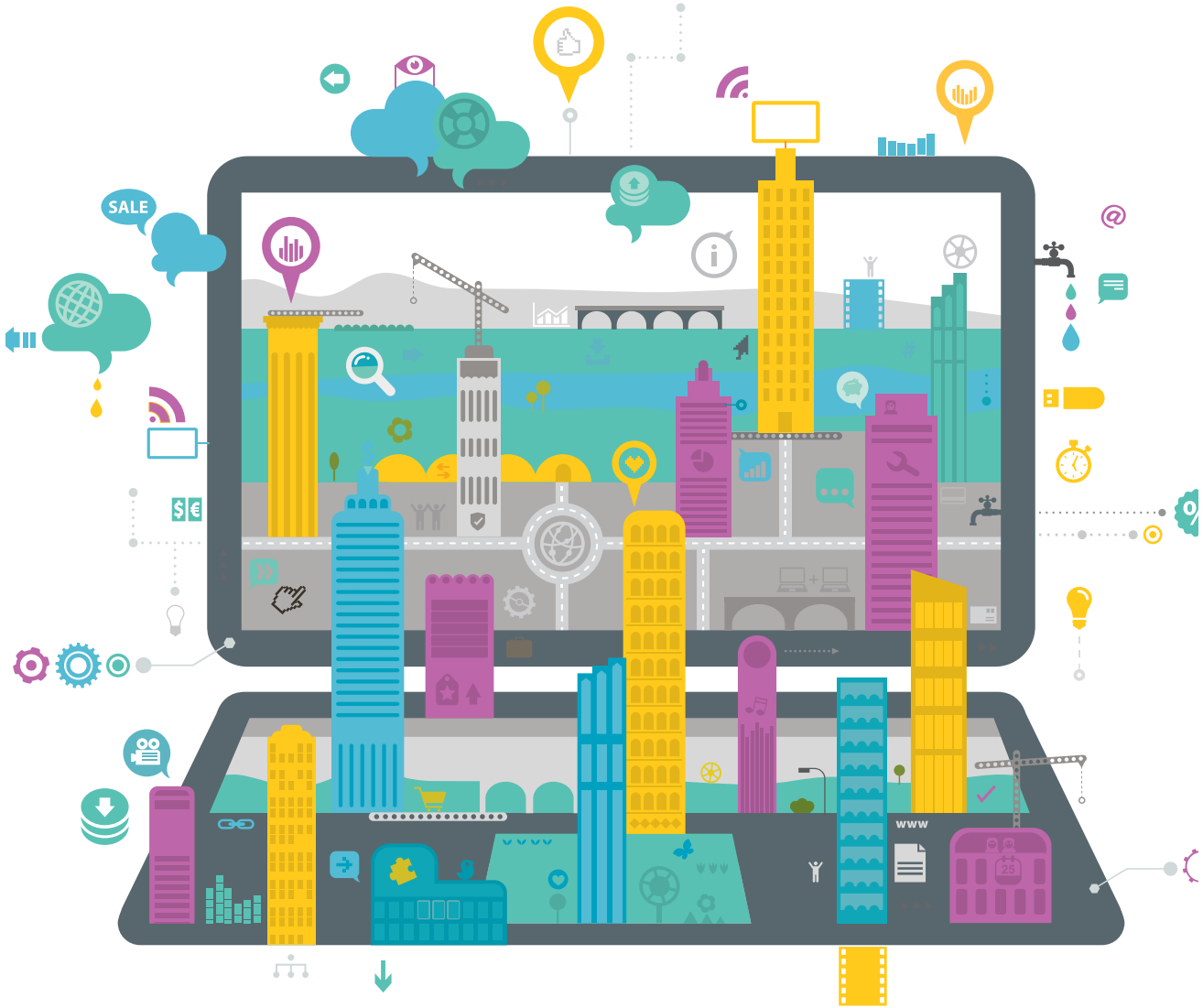
Newman argues the case that DevOps is a cultural movement, built around individuals. As such he questions how traditional certification can measure this: "I do have some respect for some sorts of certification in IT, but very little of it. I see this in the same class as making sure the person who installs my gas boiler knows their stuff."

"DevOps is a **cultural movement**, built around individuals."

He questions whether you can have faith in the DevOps abilities of someone who has a certificate of the type offered by the Institute. Indeed he ponders whether there might be an ulterior motive at work here.

"A cynic might wonder aloud whether the use of certification is a way for trainers to set up a closed shop and make more money as a result...The problem is that there is always going to be a conflict of interest when the same group that sets up the certification process controls who can deliver certified training, at the same time as benefiting financially from delivering training themselves."

He concludes: "Go ahead and make money from DevOps with my blessing (not that anyone is asking for it). Offer training, set up whatever institute you want. But please, don't claim this sort of certification is going to fix the problems we face in the industry."



James Smith at DevOpsGuys is similarly a naysayer. "I fundamentally disagree with the whole concept as it currently stands and I think the main challenge is that there's still no real agreement about what DevOps is," he says. "We have many people stating that it's not about tools, it's a philosophy, a professional movement, in which case what exactly do you certify?

"If it is about 'the cultural and professional movement that stresses communication, collaboration, integration and automation' as the DOI website says, then what credentials do the DOI have to state that their way of DevOps is the right way? There is no one-size-fits-all here."

He adds: "I also think that we are looking at the wrong things with DOI. If DevOps is underpinned by Agile, CI and CD—if it's driven on Automation, Lean, Measurement and Sharing—then as an industry we need to focus on getting the foundations right before we start certifying the outcome. I would much rather see training effort put into Automation tools, understanding of lean / agile practice and so on."

Contino's Benjamin Wootton takes a more conciliatory tone when he writes "I'm probably one of the few people in favour of something like the DevOps Institute. Perhaps this isn't perfect in its current form, but it's a step in the right direction from what I can see. Why? Because I think DevOps needs to be made more 'enterprise friendly'—packaged up for that audience and made easier for them to consume and adopt.

> "The DevOps community is big and getting bigger. There is room for a wide spectrum of opinion."

"As a concept, I think this kind of organization could help," Wootton suggests. "A 'grown-up' organization like the DevOps Institute, talking in the right language and bringing the associated training and certification, could remove some of those unfair objections, give DevOps a level of maturity, perhaps get it 'in the door' of more organizations and allow the other advocates to follow up behind with the real meat of the message."

For its part, the Institute recognizes that its existence has sparked a debate in the DevOps community. In a blog posting at DevOps.com, Editor-in-Chief Alan Shimel admits that consensus was unlikely any time soon.

"I know there is a segment of the DevOps community who view this as an apocalyptic event," he notes. "While I obviously don't agree with that view, I respect their opinion. As I have said all along, the DevOps community is big and getting bigger. There is room for a wide spectrum of opinion.

"If you don't think that DevOps needs formal education, training and certification, so be it. I and many folks I have spoken to believe that as DevOps continues to cross the chasm into mainstream enterprise IT, a formal training and certification program will be required. Whether it is from the DOI or some other entity is the only question."

For my part, I do see the need for a body advocating standards for the DevOps movement as it continues to integrate with mainstream IT. It could also be a valuable voice in support of government and training initiatives, lobbying for investment in this area—which is what I would like to see the Institute start to do. However, a stronger case for certification is needed as the current pass rate of 65 percent is fairly low. For example, the UK driving test written

(theory) exam requires 43 out 50 questions answered correctly – an 86% pass rate. A cynic might suggest that 65% is the lowest threshold to pass whilst retaining credibility and maximising the chances of those who have paid for the course passing and becoming "certified". You can have a third of the multiple choice questions wrong and still be certified; not an especially high bar.

Coming back to the driving test analogy, whilst passing the written and practical components gives people the certification to drive a car, it doesn't necessarily guarantee that they are competent drivers, nor does it ensure that they will remain good drivers for years to come. Practical application of what is learnt on the course and on the job are essential.

"It could also be a valuable voice in support of government and training initiatives."

To avoid going down a similar route and producing below-par DevOps practitioners, the DOI may need to revisit its criteria in the near future to ensure that, as DevOps becomes more a part of the mainstream IT environment, it is handled by the best. Finally, one aspect that has surprised me has been the lack of a "buzz" around the Institute, positive or negative – tweet and follower volumes have not achieved any critical mass and response overall seems to be muted. It will be interesting to revisit the Institute's progress 12 months from now and witness its take-up, feedback generated and overall evolution.

What is "Micro" about a **Microservice**?

Jason Bloomberg, President, Intellyx

One of the hottest new terms in the world of enterprise computing is the microservice. Starting with the seminal 2014 article by James Lewis and Martin Fowler of ThoughtWorks, microservices have taken on a life of their own – and as with any other overhyped term, they have generated their fair share of confusion as well.

Perhaps the best definition of microservices comes from Janakiram MSV, Principal at Janakiram & Associates. "Microservices are fine-grained units of execution. They are designed to do one thing very well," according to Janakiram. "They contain everything from the operating system, platform, framework, runtime and dependencies, packaged as one unit of execution." As a result, "a microservice architecture promotes developing and deploying applications composed of independent, autonomous, modular, self-contained units."

And yet, the above definition leaves us with yet another question: what does it mean for a unit of execution to be fine-grained? The obvious answer is small, as in micro, the prefix that gave microservices their name. But does this notion hold water?

The **challenge** of defining granularity

Unfortunately, granularity is a rather general term. Many different things can be more or less granular, or finer- or coarser-grained. The notion of granularity in the context of services arose in the early 2000s in the context of web services: contracted software interfaces that comply with a set of XML-based standards.

However, since web services were interfaces rather than software components, granularity referred to the granularity of the interface. In other words, a fine-grained service sent and/or received a small number of values (say, a single number or string), while a coarse-grained service sent and/or received structured information that contained several values (for example, an XML document), as shown in the diagram above.

Determining the appropriate granularity for a service interface was tricky, as there were pros and cons for any level of granularity. Fine-grained interfaces generally lacked business context, but tended to be more reusable. Coarse-grained interfaces, on the other hand, often had a clear business context, but were typically purpose-built for a particular situation, thus limiting their reusability.

Microservices, in contrast, are more than interfaces. They are the whole package, as Janakiram points out—a unit of execution including code, runtime, and more.

In his new book, *Building Microservices: Designing Fine-Grained Systems* (O'Reilly, February 2015), Sam Newman says, "The question I am often asked is how small is small? Giving a number of lines of code is problematic for a number of reasons, including language differences and the specifics of the task at hand".

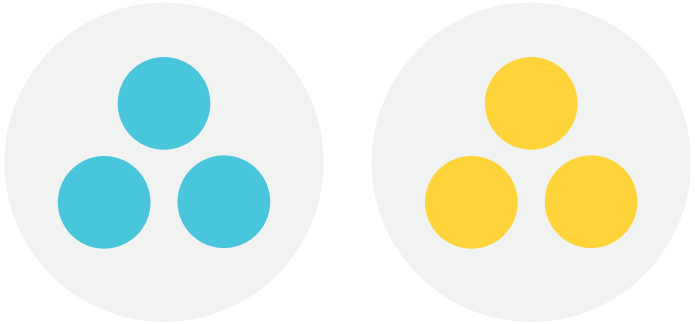
He offers a practical measure. "If the codebase is too big to be managed by a small team, looking to break it down is very sensible," Newman posits. However, he adds a caveat: "the smaller the service, the more you maximise the benefits and downsides of microservice architecture."

In other words, make your microservices too small and you'll have to manage excessively large numbers of them, but make them too big and you'll lose the benefits that drove you to create microservices in the first place.

Newman makes one other comment on the subject: "Another somewhat trite answer I can give is small enough and no smaller." Trite, perhaps, but this point underscores an important principle of microservice construction: parsimony.

A parsimonious microservice is as small as it should be – and as Newman says, no smaller. In other words, during your iterative refactoring efforts (part of any Agile approach), revisit your microservices and see if there's anything extra in them, or if splitting up a microservice into two or more microservices improves matters. If so, continue to pare them down and split them up until such efforts no longer move your project forward.

microservice with
low cohesion



microservices with
high cohesion

Revisiting software **cohesion**

Another important microservices principle – and in fact, a principle of modular software design since the 1960s – is cohesion. A microservice is highly cohesive if its elements belong together – in other words, it does one thing and does it well, as shown in the diagram above.

Web Services frequently suffered from low cohesion. Sometimes WSDL files contained dozens or even hundreds of operations, where a single service did a wide range of different tasks. Reacting to the problems with that kind of service is one of the primary motivations for microservices.

Therefore, while parsimony and cohesion make more sense in the context of microservices, granularity makes more sense in reference to interfaces – and the concepts are quite different. After all, a well-built microservice might (and typically would) have a coarse-grained interface, for example, if it accepted and/or returned a JSON document.

> "Focus on the parsimony and cohesion of your microservices, and the granularity of their interfaces."

However, as the title of Newman's book suggests, an entire system built with microservices would itself be fine-grained – yet another sense of the notion of granularity. A collection of microservices would be fine-grained if those microservices tended to be parsimonious and highly cohesive—regardless of how big they were, and even though their interfaces should be coarse-grained.

If it were up to me, I wouldn't refer to microservices or systems of microservices by their levels of granularity at all – but unfortunately, it's not my call. So heed this warning: granularity is a slippery concept. Focus instead on the parsimony and cohesion of your microservices, and the granularity of their interfaces.

Copyright © Intellyx LLC.

Flipping **incentivization**

James Smith, Co-Founder, DevOpsGuys

How should DevOps be incentivized? It's a question that organizations increasingly find themselves asking. James Smith, one half of DevOpsGuys, has over 15 years' experience delivering and managing enterprise web applications for global blue chip companies.

Today, DevOpsGuys is at the forefront of technology promoting Lean IT, DevOps and Continuous Delivery across the enterprise to ensure alignment with commercial opportunities, speed-to-market and client impact. Based on that experience, Smith perceives a need to rethink some roles:

"Typically there are a bunch of incentives for DevOps which are based around driving behavior," he says, "but we've been focusing on the wrong things."

Smith lists numerous examples of inappropriate incentivization in DevOps circles. For example, in some organizations, developers are rewarded for the number of bugs that they fix, but this can simply lead to the release of buggy code: the more bugs released, the more there is to fix.

Then you've got the so-called 'death march', where DevOps people are incentivized by the number of hours that they work. In these cases it makes little financial sense to finish as quickly as possible when more hours = more money; something that should take one hour suddenly takes ten.

> "The more bugs released, the more there is to fix."

Or there's the expert incentive, where someone is given ownership of a piece of technology and becomes, effectively, the single point of failure. What can happen next is those people end up becoming the only ones who understand the situation completely. The organization ends up incentivizing 'knowledge hoarding' over the sharing and distribution of information.

Role **reversal**

Smith argues for the re-examination of roles and their responsibilities and the incentives typically applied to them. "If you're in development, then you are incentivized to change things. You're all about developing new features, changing existing functionality and so on," he says.

"So you're incentivized for speed and risk: the faster you can get stuff out, the better. But if you're in operations then you're largely incentivized to reduce change, to maintain stability, certainty and predictability."

That's the traditional model for organizations, Smith believes. To break the mold and better align incentives across the teams, swap the roles around so that you incentivize the development team on stability, uptime and managing certainty, and incentivize operations on value creation and how well they innovate and manage change.

"You want to make sure that both sides are carefully aligned and understand what the other does," he suggests. "You can put each group in the other's shoes. You have them sitting on the other side of fence.

"Take the development team and put them on call and make them responsible for the software. They take ownership of front-line issues, dealing with the support tickets, which should give them an idea of the supportability of their software and a greater understanding of what they're producing.

"In the same vein, the ops guys are pushed into the development process, so they get an understanding of the best practices of managing change, having to use source control, peer-reviewing code, looking at automation techniques and so on. This should give them an idea of how they can enable change quickly in their organization."

Having this shared perception of one another's roles enables a realignment of goals around both groups, but in this process it's important to bear in mind that the one goal absolutely not up for discussion or compromise is quality. "Both teams need to be centrally aligned around customer value and quality," cautions Smith. "When the goals are aligned, the rate of delivery and the quality of the product increase."

All of this definitely needs senior stakeholder support within the organization and you're almost certainly going to run into some ground-level resistance to change. Smith notes: "People are creatures of habit. They're brought in

"When the goals are aligned, the rate of delivery and the quality of the product increase."

to do a role, then you ask them to do something else and they say 'That's not in my employment contract'.

"You'll find that in some companies there is resistance of that nature, but it's really fear of the unknown, so you need to show the benefits and help them to see that the changes are going to help them to do their jobs better."

Ration these role-reversal periods as well. "Initially this should be done for short periods of time," says Smith. "But the frequency is important. It shouldn't be a one-off exercise."

It's also essential to be able to measure the impact of the role changes. "You can start to encourage new behaviors, but unless you measure the impact you're just dealing with opinions, not facts," warns Smith. "Get measurements around the rate of change or get quality metrics or whatever you want to measure. But do put some kind of measurements in place."

Overall you need to be able to show that adjusting the traditional incentivization practices helps DevOps people to do their jobs better and contributes to the benefit of the organization.

"At the end of the day, you are improving your application quality. You are improving the products that you are producing," concludes Smith. "There's a strong correlation between creating high-performing IT teams who are able to deploy products more frequently and keep uptime at a higher level and contribution to the bottom line."

Educating **DevOps**

Em Campbell-Pretty, Partner, Context Matters

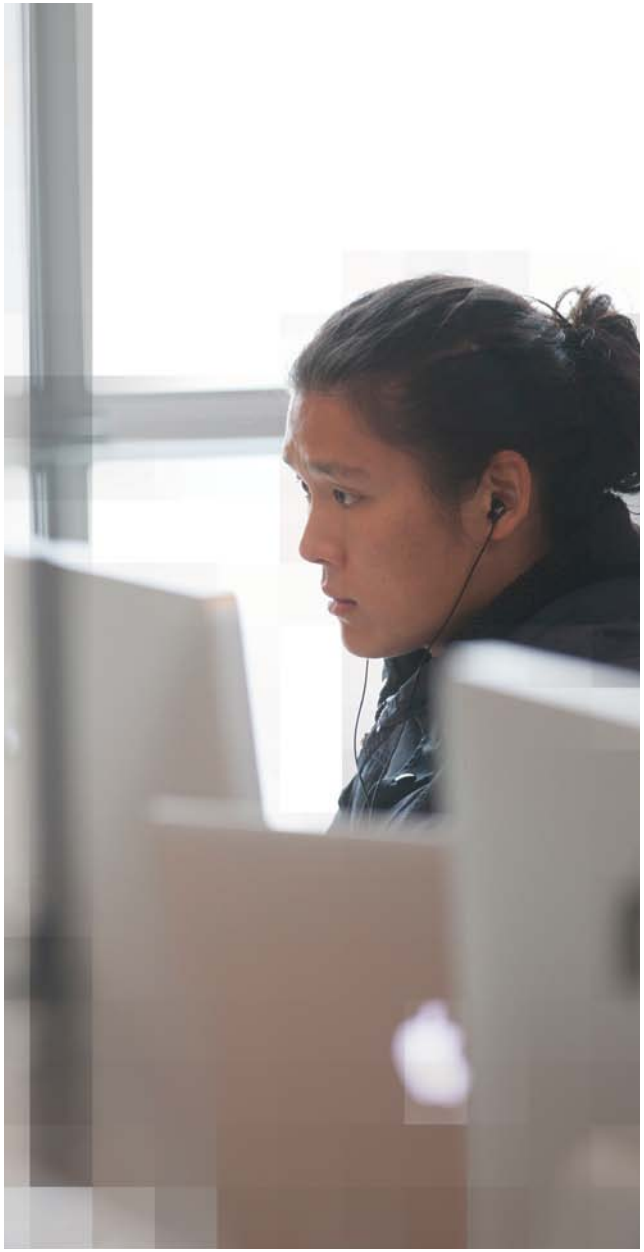
In a career spanning more than 20 years in business, Em Campbell-Pretty, Partner at Australia's leading enterprise Agile consultancy Context Matters, has witnessed first-hand some of the tensions involved in that long-standing dilemma: how to bridge the gap between business and IT.

From a business perspective, she's had good and productive engagements with IT counterparts, but equally she's had experiences that were less so, at times bordering on the hostile. After discovering Agile, she became a passionate advocate of its ability to align business and IT around the delivery of value and of the potential of introducing DevOps to enterprise organizations.

The first thing the business decision-maker needs to do, she advises, is get informed and assume responsibility for your own education. Don't assume you're going to get all the information that you need unless you ask for it.

She says. "I spent a very long time as a business person and as a business person you just assume that the IT folk know what they're doing. Furthermore, we tend to be conditioned not to question that assumption.

"When I started learning about DevOps and Agile, that was when I started asking questions. You really do need to ask questions. You need to read the books and attend the conferences and ask the questions because that is how you move from the assumption that all is well with the technical practices within your organization to a more realistic understanding of the true state of play."



Educate yourself

Campbell-Pretty is a firm believer that it's incumbent on business people to get up to speed, if only to put themselves on a level playing field with IT. "I had an Agile coach who believed that people should educate themselves. That pushed me to focus my reading time on Agile classics like Jim Highsmith's Agile Project Management," she explains. "It's just so, so powerful just to take the time and investigate.

"It's not easy to get people to read, but those who do are far more successful. I recently heard Verne Harnish, author of Scaling Up, speak at a conference and he made the point that 'those who can read, and don't, are only marginally better off than those who can't' and he's right," she adds.

"If nothing else, reading up enables you to ask questions of the right people. I remember asking IT what the automated test strategy we had was and getting a blank expression back. The question hadn't been expected. The IT people didn't like that much."

That blank look reminds us that there's obviously a danger that challenging IT on their home turf might not be welcomed, but it's worth it, reckons Campbell-Pretty.

"It doesn't always go down particularly well with IT when the business starts asking questions," she admits. "But by the same token the IT folk don't always take time to explain everything to the business."

Campbell-Pretty has personal examples that illustrate her point. "The first time I picked up a book on Agile, it was because I'd had a conversation with an IT project manager who bombarded me with jargon, she recalls. "The project manager came to me and said that she intended to deliver the project using Scrum, which would involve locking people in a room for a month, at the end of which they would emerge with working software. My head was spinning. What nonsense were these people talking about? So I logged on to Amazon and ordered some books on Agile.

"A couple of years later, the role of business sponsor for that particular project was transitioned to me. The project had not ended up using Agile or Scrum and delivery was not going well. We were spending a lot of money and not getting business outcomes. There had to be a better way. This brought me full circle back to the reading I had done about Agile and eventually the entire program being transitioned to use Agile and later including DevOps."

CIO – help or hindrance?

But if the business side is getting armed with the right questions to ask, the next question is how far CIOs are themselves up to speed with the answers, particularly around DevOps. Campbell-Pretty is uncertain on this point.

"It's hard to generalize if CIOs get DevOps. I don't think it has gone mainstream yet," she suggests. "There are very big companies out there which are playing with DevOps, although it seems to be mostly in the digital space. Taking on DevOps in large companies with big, heavy, legacy applications is a very different challenge. I'm not sure how many enterprises are facing into the DevOps challenge at the moment."

In fact, Campbell-Pretty makes the case that CIO involvement may not always be entirely helpful to the cause of successful DevOps introduction to an enterprise.

"What tends to happen when the CIO gets involved is that the organization sets up a DevOps Centre of Excellence," she argues. "Frankly I'm not sure that I buy into that as a mechanism for rolling out DevOps because it becomes a top-down mandate. What you need to do is to change the culture and that's just not the same thing as someone swooping in from head office and saying 'Do what I say'. It's in no way as powerful as a ground-up approach to DevOps.

> "Do what I say' is in no way as powerful as a ground-up approach to DevOps."

"You do need funding and expertise and those will likely come from a central source, but when these things become CIO-driven, you can find yourself with a problem. If you're going to try to solve the DevOps problem in large organizations through a central mandate, then you're going to miss out on huge opportunities for people to get better and better by improving what they do day in and day out. You need to create tribes and harness their energy to inspect and adapt and innovate."

At the end of the day, it's essential to get those business/IT conversations happening for mutual benefit—and that involves both parties upping their mutual respect. "We have a problem that there is a lack of business people who respect technology people and vice versa," she concludes. "The right conversations just don't happen enough.

"Business people are pretty bright you know. They run million-dollar businesses. So IT really can go to business people and talk to them in science-based and fact-based terms. Help them understand why it makes sense from a business perspective to invest in DevOps. Then they will get on board and potentially even become your greatest ally in your journey to DevOps. "

Wielding the double-edged sword of **automation**

Dan North, Dan North Associates Ltd

When delivering software, automation seems like a really cool trick. You can easily show a manual release or build, do some automation 'magic' to show that the build is now automated, put metrics on it, show it is more efficient and deliver huge time savings. Doing that can be very popular.

However, before automating it is essential to consider its opportunity cost, the cost of everything else you could be doing instead, and whether automation is even appropriate. Specifically, is automation being applied to the right things, at the right time in the project?

Choosing what to **automate**

If you have a deterministic transformation, such as compiling code to binary, where all of the stages are tightly defined, this is clearly a good candidate for automation. This is why hand-compiling code is a minority sport now.

However, automation becomes a double-edged sword when it automates a process or activity, as these tend to be a function of context. If that context drifts, then the automation solution may no longer be appropriate, and may even be detrimental.

For example, with cars it currently makes sense to automate transmission but not steering, as steering is a function of the behavior of other cars on the roads. Although, as Google and others have shown, you can manage to automate steering, even around ducks, it is just not currently economic to do so.

The choice over braking automation is, however, more clear-cut. You can teach drivers that in an emergency, cadence braking will let them steer and slow at the same time, but the reality is that in an emergency most drivers will fail to do so. An ABS braking system will 'remember' to cadence brake, and do so more effectively, every time. This makes applying automation here deterministically a better solution than the manual alternative.

The early stages of a project are when you know the least about everything—the organization, the technology used, the operations environment, the constraints and your team—yet this is often when people reach for that automation 'magic', seeking speed and repeatability. As soon as the team has what looks like working code, they automate the build. Unfortunately, this crystallizes the current knowledge of the project, including all your erroneous assumptions. Any errors in understanding of how things work are baked in, and the chances are they will stay there and never be reviewed. They remain fixed by the 'we have features to ship' mentality, no matter how slow or complex the build is, it is THE build and there is always something more pressing to deal with.

> "automation needs to be challenged and adapted."

One of the core principles of Agile is adapting to change over following a plan. So automation, which is mechanistically following a plan, needs to be challenged and adapted as part of Agile working. Go back to manually building if necessary, question the sequence of events, evaluate options for parallelization, try running slow processes first, test whether the impossible can happen, as what was once best fit is often no longer optimal.

The **opportunity** cost of automation

Visiting a team working for a financial services client, I found them investing enormous effort constructing their build pipeline. It was a thing of beauty, with automated testing stages, reporting and all kinds of fancy instrumentation. I visited them around six weeks into the project and I asked what demonstrable client features they had delivered. They looked embarrassed—nothing had gone through that pipe.

The conversation with the client was equally uncomfortable. They had bought into the benefits of automation but to their stakeholders this was looking anything but agile.

Getting the process of automation right requires considering what you are not doing while you are automating, the opportunity cost and where the value is for the client.

Valuable **work**

So how do you discover where automation will add the most value? I advocate recognizing that there are different types of work that are valuable to a project.

Agile methods like Scrum tend to only explicitly recognize one type of work, which is delivery of features. Fundamentally, all Scrum measurements are features metrics: velocity, story points, burn-up, burn-down. As well as feature delivery, I believe teams should also value two other types of work as first-class citizens, namely discovery or exploratory work and Kaizen.

Discovery work, understanding more about the problem you are trying to solve, is first-class work. It happens anyway, it is what people do, but it is not explicitly recognized and so is undervalued. Active discovery can be the key to a much shorter path through the problem. Maybe you can get the same business outcome with fewer, different features.

Kaizen in this context is not just confined to the narrow definition of continuous incremental improvement, it encompasses improving the system in which you are operating. Knowledge transfer is a great example of applying Kaizen to the delivery system. If one team member has a skill and teaches that skill to another, capacity for that work is doubled. If tacit knowledge is documented, it becomes available to all the team, which increases their capacity to solve problems, so we have made a better system for delivery.

Done well, automation is a form of Kaizen. It improves the development process, speeds testing and reduces likelihood of defects by eliminating manual work. But since Kaizen is rarely recognized as first-class work it gets done 'in the cracks' so it is not governed, not subject to the due diligence and oversight of delivery work. There are small, tactical instances of explicit Kaizen activities, such as during a 'sprint zero', where the build gets automated, but as discussed earlier this is often the wrong time.

> "Boring is a necessary, but not sufficient, condition for automation."

So Kaizen gets hidden. Developers instinctively know it is useful, and indeed necessary, to sharpen the axe, but with no 'official' sanction to hang the work on they end up flying under the radar to do it anyway.

Investing in discovery and Kaizen work is an example of having made the choice between hacking and strategic development. It needs to be elevated to first-class work and made visible and demonstrable.

Demonstrating delivery is simple: you can demo a new feature, showing something that was not there before.

The only way to 'measure' discovery or research activities is by time-bounding. Allocating time to dig into a particular facet of the project, running experiments, reviewing data, then asking the question: is there more to be gained through additional research, will we progress in this area, or are returns diminishing?

We can demonstrate knowledge transfer using a show-and-tell by the newly trained team member, or by having them document their learning.

Kaizen for the delivery process can be measured in terms of how much time will be saved next month by automation, e.g. time not spent debugging or diagnosing flaky manual deployments, and can be assigned a value just like features.

Seek boredom and **customer** value

Experience has moved me from 'If in doubt, automate' to 'Don't automate until something is *boring* (and even then maybe not)'.

Boring is a necessary, but not sufficient, condition for automation. If a process is boring it means you have done it often enough that you know how it works, and that it has become repeatable enough that surprises are unlikely. When you have both of these conditions, you have a candidate for automation.

With that candidate lined up, you then need to think like your customer, and assess the relative value of investing in each type of first-class work.

Apply this discipline and the double-edged sword of automation can cut you free you from the boring stuff and let you carve through what matters.

Tony Chapman

Managing Director,
LinuxRecruit

Tony Chapman has founded and built specialist DevOps agency LinuxRecruit who are working with organisations across the UK, designing their DevOps recruitment strategy and fully staffing their DevOps Engineering teams. Tony has been working in the Open Source community for 10 years and was recently shortlisted for Recruiter of the Year 2012 at the prestigious national Recruiter for Excellence Awards, LinuxRecruit were shortlisted for newcomer agency of the year at the 2013 awards. He is a contributor to the Open Source community, has a regular column in Linux Format Magazine and is co-organiser of the world's biggest monthly DevOps meetup, the DevOps Exchange in London.

Patrick Hyland

Founder, DevOps Associates

Patrick Hyland is the founder of DevOps Associates, a London-based consultancy concerned with application engineering management. The consultancy applies a blend of agile methods, connected ITIL lifecycle processes and DevOps collaboration/engineering practices to help companies design, build, deliver and operate outstanding application services.

Patrick is an ITIL expert with 18 years of development and operations experience. He is particularly interested in management via Eli Goldratt's theory of constraints, applying a lean manufacturing mindset within an IT Service Management context.

Grant Smith

Author, Next Gen DevOps

Grant has driven real collaboration between Operations and Development teams in AOL, Electronic Arts and British Gas by implementing Infrastructure as code and driving application integration from continuous build systems. Grant has delivered game platforms running in the cloud enjoyed by millions of players per day and websites serving a billion page views per month. Most recently he has delivered a high performance, scalable Internet-of-things platform for British Gas. Grant is the author of Next Gen DevOps: Creating the DevOps Organisation and is frequently sought out for his cloud and DevOps expertise. Grant can be reached at grant@nextgendevops.com

Nicole Forsgren

Director of Organizational Performance and Analytics, Chef

Nicole is the Director of Organisational Performance & Analytics at Chef and an Assistant Professor of MIS and Accounting at the Huntsman School of Business at Utah State University. She received her PhD in Management Information Systems and her Masters in Accounting from the University of Arizona. She is an expert in IT use, DevOps impacts, and communication and knowledge management practices, particularly among technical professionals. Her background spans analytics, enterprise storage (specialising in RAID performance), cost allocation, user experience, and systems design and development. She is a featured speaker at industry and academic events and is involved in women in technology initiatives.

Matthew Skelton

Co-founder and Principal Consultant, Skelton Thatcher Consulting Ltd

Matthew Skelton has been building, deploying, and operating commercial software systems since 1998. Co-founder and Principal Consultant at Skelton Thatcher Consulting, he specialises in helping organisations to adopt and sustain good practices for building and operating software systems: Continuous Delivery, DevOps, aspects of ITIL, and software operability.

Matthew founded and leads the 1000-member London Continuous Delivery meet-up group, and instigated the first conference in Europe dedicated to Continuous Delivery, PIPELINE Conference. He also co-facilitates the popular Experience DevOps workshop series and is co-editor of Build Quality In, a book of Continuous Delivery and DevOps experience reports.

Skelton Thatcher Consulting

Justin Vaughan-Brown

Global Digital Transformation Lead, CA Technologies

Justin Vaughan-Brown is Global Digital Transformation Lead, Product Marketing at CA Technologies. He is the author of 'The Digital Transformation Journey: Key Technology Considerations' paper, hosts the quarterly DevOps Influencer Dinners and is responsible for the DevOps Simulation Experience, an interactive online workshop that explains core DevOps principles.

Jason Bloomberg
President, Intellyx

Jason Bloomberg is the leading industry analyst and expert on achieving agile digital transformation by architecting business agility in the enterprise. He writes for Forbes, Wired, and his biweekly newsletter, the Cortex. As president of Intellyx, he advises business executives on their digital transformation initiatives, trains architecture teams on Agile Architecture, and helps technology vendors and service providers communicate their agility stories. His latest book is The Agile Architecture Revolution (Wiley, 2013).

James Smith
Co-Founder, DevOpsGuys

James, co-founder of DevOpsGuys, has over 15 years' experience delivering and managing enterprise web applications for global blue chip companies. Today, DevOpsGuys are at the forefront of technology promoting Lean IT, DevOps and Continuous Delivery across the enterprise to ensure alignment with commercial opportunities, speed-to-market and client impact.

Em Campbell-Pretty
Partner, Context Matters

Em is a Partner at Context Matters, Australia's leading Enterprise Agile consultancy. After close to 20 years in business management roles within multinational blue chip corporations, Em discovered Agile and became passionate about the chance it provides to align business and IT around the delivery of value. In 2012, she launched Australia's first Scaled Agile Framework (SAFe) Agile Release Train. Em is an active member of the global agile community and was invited to co-chair the Enterprise Agile track for the Agile Alliance conferences in 2014 and 2015. Em also blogs about her "Adventures in Scaling Agile" at PrettyAgile.com. Em can be contacted at em@contextmatters.com.au

Dan North

Dan North Associates Ltd

Dan North uses his deep technical and organisational knowledge to help CIOs, business and software teams to deliver quickly and successfully. He puts people first and finds simple, pragmatic solutions to business and technical problems, often using lean and agile techniques. With over twenty years of experience in IT, Dan is a frequent speaker at technology conferences worldwide. The originator of Behaviour-Driven Development (BDD) and Deliberate Discovery, Dan has published feature articles in numerous software and business publications, and contributed to The RSpec Book: Behaviour Driven Development with RSpec, Cucumber, and Friends and 97 Things Every Programmer Should Know: Collective Wisdom from the Experts. He occasionally blogs at http://dannorth.net/blog.

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.

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

#BusinessReWrittenBySoftware

Contributions and comments were solicited following a discussion held in March 2015 and in subsequent email interviews.
Copyright ©2015 CA and its licensors. All rights reserved. All trademarks, trade names, service marks and logos referenced herein belong to their respective companies.