Executive Insight Paper

In association with

Taking DevOps to the Next Level

It's all about continuous delivery

June 2016

Introduction

The penny drops on the DevOps scalability debate

Many appreciate the potential of DevOps, but uncertainty persists about how well it scales in a larger enterprise environment.

A lot can be learned from getting a group of experienced CIOs in a room and encouraging them to talk freely about the topics at the front of their minds. Rarely do they want to discuss technology, and even the latest IT industry 'hot topics' usually don't get that much airplay. Conversations, and sometimes quite heated debates, tend to revolve around broader business-related trends, how these are changing stakeholder and user behaviour and expectations, and how best to respond to such dynamics.

At a recent CIO roundtable, however, being run under the Chatham House rule (so participants could speak openly and remain confident about anonymity), the group got very animated about one particular hot topic.

It started with a debate around the skills mix required to deal with new and changing demands, and part way through, the CIO of a medium-sized 'born on the Web' company said "It sounds as if you guys are not set up for DevOps." Her observation was met by some mumbling and shuffling among the group, most of whom had IT teams many times the size of hers. The CIO of a big utilities organisation said that it probably wouldn't work in their highly regulated environment. Others made comments about some of their teams 'doing DevOps' on certain projects, but they were uncertain how it could scale.

Success at scale requires building the right process and tooling framework to enable 'continuous delivery'.

At this point, the CIO of a large international financial institution stepped in and said "Well we have transformed a large part of our IT delivery through taking DevOps principles on board." He went on to say that it was still a work in progress, partly because the required cultural change takes time, but also because they didn't necessarily do all the right things up front. But they were now delivering solutions and features a lot more efficiently and reliably, and at least an order of magnitude more quickly. This got everyone's attention.

After further discussion of the practicalities and benefits, you could hear the proverbial penny drop in the minds of many people present. One then asked what the CIO would do differently if he could wind the clock back and start his 3-year DevOps journey over. His response was "Get the stakeholders on board a lot earlier." He made the point that it's impossible to implement DevOps purely within IT – the business needs to switch mindset too. The iterative delivery of the value and the experimental and continuous improvement philosophy means thinking differently about everything, from specifying projects and requirements, through allocating and managing budgets, to ultimately measuring success.

Investment will be needed, and an ability to articulate the benefits of continuous delivery is essential to making the business case.

Another big reason cited for getting stakeholders involved sooner is because you can't scale DevOps in a sustainable way without investing, which in turn depends on gaining the support of those who care how IT budget is spent. Expanding on this, our CIO explained "A lot of DevOps is about culture and the way you organise and motivate your teams, but that's not enough. You also need to deal with the science and engineering part of the equation." What he was referring to was that in order to scale DevOps, you need to create a robust framework of tools and processes to enable 'continuous delivery'. As he spoke about this very tangible aspect of DevOps, yet more pennies could be heard to drop.

In the remainder of the paper we'll explore what 'continuous delivery' means, take a high level look at what you need to implement it, and identify the key elements of a case for investment in it. But first let's take a minute to sanity check the business need.

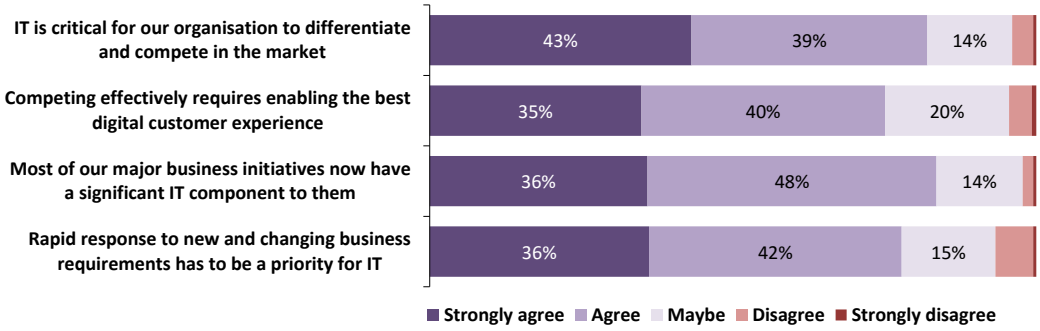
Digital business requirements

Faster, experimental and more customer-driven delivery

A recent study of over 400 senior IT professionals (Appendix A) provided a clear picture of what's driving the demand for IT teams to deliver better and faster (Figure 1).

Figure 1

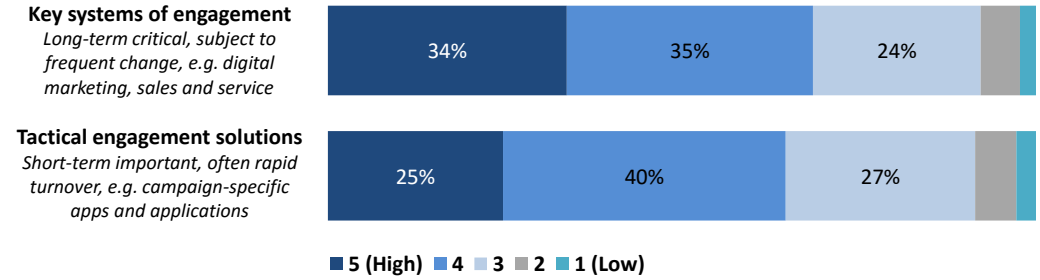
How much would you agree or disagree with these statements?



Rapid and continuous innovation is the underlying imperative here. This is reinforced when we look at the attention that's nowadays often paid by business stakeholders to so-called 'systems of engagement'. What's notable about this is that it's not just about your core marketing, sales and customer service solutions. The growing need for applications of a more tactical nature also comes through very strongly (Figure 2).

Figure 2

What level of attention do business stakeholders have on the following categories of application?



This focus on tactical engagement is an indicator of the need to experiment in today's digital world. The idea is to get capability based on new ideas into the hands of customers very quickly, then assess how well it performs and proceed from there accordingly. If it works, you can refine it or extend it with new features, and build on the success in an iterative manner. Incremental value is delivered early and often, and you can be sure you are always headed along a path that will meet both customer and business expectations. If you don't get the result you were looking for with the initial solution, and modifying it seems unlikely to succeed, then you can kill it and move on to the next idea or try something different to meet the objective. Some refer to this as 'fail fast'.

The idea is to get capability based on new ideas into the hands of customers very quickly.

The reality, of course, is that this kind of approach applies equally when considering the evolution of core engagement systems. Here the focus is on which additional features and functions are going to drive incremental value, but the principle is the same.

Either way, everything we are talking about here is in stark contrast to the traditional project delivery model. It's a very big departure from a world in which the full scope of a solution is defined and committed to upfront, then a complete system is deployed after months or even years of development, integration and testing activity. When requirements and expectations represent a constantly moving target, which is the reality of digital engagement systems, the problems are obvious.

Achieving speed and flexibility - safely

From Agile to DevOps to Continuous Delivery

All too often, the rapid flow of output from agile projects builds up behind the dam of operational handover.

It could be argued that many of the needs we have been discussing are fulfilled by agile development, but this only deals with the upstream part of the equation. All too often, the rapid flow of output from agile projects just builds up behind the dam of operational handover. Creating a stable software build is one thing, installing it into even a test or staging environment, let alone getting it into production, is another. A lot needs to happen in terms of platform provisioning and configuration, for example, and making sure that software is deployed in a controlled and manageable manner. This is the domain of the traditional operations team, who over the years have put barriers and controls in place to make sure that new software or updates can move into the live environment without undermining service levels or compromising security and compliance.

This protective approach has historically made sense when you consider the broader set of imperatives associated with IT delivery, which includes things like:

- Responding quickly/effectively to new needs
- Maintaining operational service levels
- Managing IT and information-related risks
- Controlling the cost of IT delivery and ops
- Delivering overall value to the business

Moving to a DevOps approach is the first step in resolving the conflict between speed of delivery and operational needs.

So how do you address all of these items simultaneously, given that responding quickly would seem to be at odds with managing service levels, risks and costs?

The first step is to break down the traditional barriers between Dev and Ops and get the two working in a more aligned and harmonious manner. This means moving to an integrated DevOps approach, which translates to a range of specific actions and practices. We don't have the space to go into all of these here, but if you want to know more, please seek out our paper entitled "Assembling the DevOps Jigsaw", which goes into some detail.

In the meantime, if we group the respondents in our latest study according to how well they are doing in relation to the above 5 imperatives (see Appendix A), then it's clear that High Achievers in particular seem to be driving down the DevOps path (Figure 3).

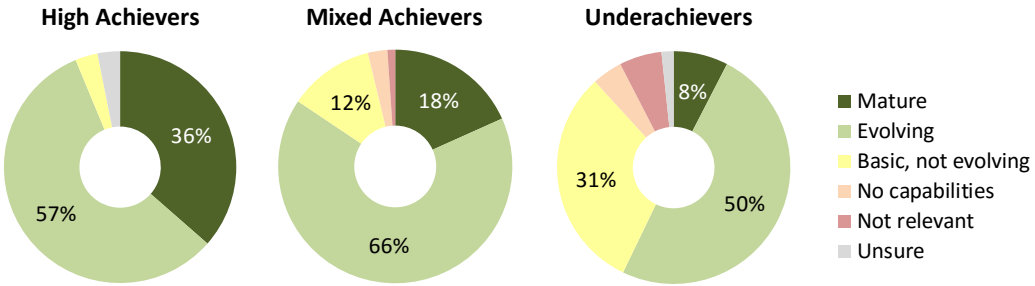


Figure 3

How would you describe your capabilities in relation to DevOps?

You will ultimately need industrialised continuous delivery.

DevOps, however, is often embraced along with the spirit of practitioner freedom. This is understandable if you view it as simply the downstream extension of agile, but it's why many run into scalability problems. The reality is that you can only get so far with ad hoc adoption of open source tools and point solutions. At some point you need to move onto the next level and create a more industrialised continuous delivery framework.

Continuous delivery fundamentals

Automation and orchestration across the entire delivery pipeline

Ask yourself what would be involved in getting a change to a single line of code deployed into production.

A good way to focus your mind on what continuous delivery is all about is to ask yourself what would be involved in getting a change to a single line of code deployed into production. Start by thinking through the various steps, from creating a new software build, through executing the necessary tests at unit, function, integration and performance level (moving the build between environments as necessary), then ultimately releasing the change into the live environment. Now consider how much of that activity is dependent on manual processing, handcrafted scripts and/or handoffs between teams based on tickets, emails, instant messages, etc. Finally consider all of the negotiation that goes on along the way, e.g. to persuade that operations specialist to drop what they are doing and focus on getting the new build onto the relevant servers.

You would not be unusual if you felt this hypothetical scenario was simply unrealistic and therefore pointless to consider because the whole process would take days or even weeks. You might highlight that a single line change would generally have to wait for incorporation into the next big release to make all of the time and overhead worthwhile. On those occasions when you have to rush out a critical bug fix or a must-have feature to support a high-profile marketing launch, stress levels rise, overtime is worked, and personal and family lives are disrupted. No one is going to proactively invite such grief.

Continuous delivery is about introducing automation across the entire software delivery pipeline.

In a nutshell, continuous delivery is about removing all of this overhead, distraction, delay and risk. It does this by introducing automation across the entire software delivery pipeline so that every activity is performed in a fast, efficient and repeatable way. Some of the key capabilities needed for continuous delivery were highlighted in the research study mentioned earlier, and it's no coincidence that High Achievers are significantly ahead of the curve when it comes to implementing them (Figure 4).

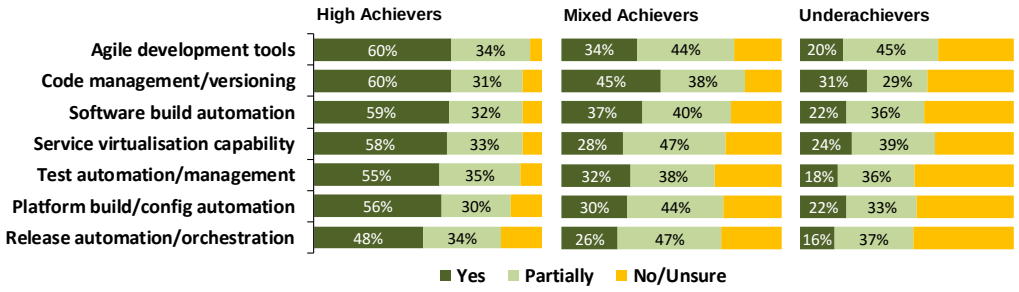


Figure 4

Do you have any of these capabilities in place to support your delivery process?

We will look at how such capabilities work together in practice very shortly. Before doing that, however, it's worth reinforcing the importance of speed, efficiency and repeatability, as these are what make the continuous flow of incremental value in relatively small units both feasible and safe. Apart from dramatically reducing cycle times, if something fails at any point in the pipeline, the process can be restarted at the relevant step following remedial action, with very little time and overhead penalty. Builds can be reinitiated, tests rerun, platforms and data refreshed, and so on, all at the press of a button. If a release automation system is in place, even button presses can be eliminated in many scenarios. Failures, conflicts, mismatches and other integrity issues can be trapped, rolled back, retried, etc., fully automatically according to predefined orchestration rules and flows.

Step-changes in speed, efficiency and repeatability allow the continuous flow of incremental value.

If all this sounds like magic, rest assured it isn't. Let's take a closer look.

Creating a continuous delivery framework

From piecemeal tooling to a properly integrated environment

Release automation is arguably the linchpin of any continuous delivery framework.

To give you an idea of what a scalable continuous delivery framework looks like, let's examine how one of the players in the market, CA Technologies (CA), addresses some of the key requirements. In the interests of full disclosure, we are using CA as a reference as it was kind enough to sponsor this paper. It's not the only vendor of continuous delivery solutions, however, and its inclusion here should not be construed as Freeform Dynamics endorsing any particular offering. That said, CA has one of the most comprehensive portfolios on the market in the continuous delivery space, so it serves as a good example.

In order to put its product portfolio into perspective, CA has coined the acronym 'RAPID', which speaks to some of the important capabilities required for continuous delivery:

Release planning and automation

This function is arguably the linchpin of any continuous delivery framework, and CA's solution here is 'CA Release Automation'. It enables you to specify the activities and flows involved in software delivery pipelines, then orchestrates activity in an automated manner from build to deployment. Key to scalability, it supports planning and execution across tens or hundreds of potentially interdependent delivery streams.

An important principle is that components should be substitutable, so integration with the broader ecosystem of open source and other tools is essential.

Agile development and testing

Agile goes hand-in-hand with continuous delivery, and 'CA Agile Central' provides the core of what's needed to manage agile projects, from the capture of user stories through to monitoring and tracking of sprints. A companion solution, 'CA Agile Requirements Designer', works in tandem with this to determine the fewest number of tests needed for full code coverage. Automated test execution is then achieved via integration with 'CA Release Automation' and 'CA Application Test'.

Provisioning of test data on demand

A number of well-established tools exist, such as Puppet and Chef, for provisioning application platforms. 'CA Test Data Management' complements these by enabling the automated, rapid and efficient provisioning and refresh of synthetic test data in line with application profiles and test case requirements. This introduces automation and repeatability into an area typically reliant on hand-crafted scripts, and avoids the security and compliance risks of working with live data extracts.

Integration with feedback-driven ecosystem

The above mention of third-party platform provisioning tools introduces an important aspect of CA's design philosophy, and a general lesson for the building of any continuous delivery framework. The principle is that components should be substitutable, so you can optimise the mix based on your organisation's requirements, or even the needs of individual projects. CA addresses this by listening to customers and building integrations with popular open source and commercial tools.

Deployment of simulated environments

It's generally not possible for the developer's desktop and test systems to contain everything that's in the production environment, especially when the software being developed has significant dependencies on other applications. 'CA Service Virtualization' works around this by simulating the systems, environments, API's and associated functionality of other applications in the form of virtual services. As part of this, 'CA Test Data Management' can be used to refresh the data that drives simulated activity.

CA aims to deliver a coherent set of tooling to form the core of your continuous delivery environment.

The idea of RAPID and the associated CA toolset is to provide a solid set of core functionality that will integrate seamlessly with other components, including (where appropriate) open source and other tools that have already been adopted by your

The objective is for activities and data to flow freely through a properly integrated framework in a highly coordinated manner.

developers, testers and operations staff. You may, for example, use selected parts of the CA portfolio together with the solutions of your choice in areas such as source code management and version control, continuous integration tooling, binary repositories, platform provisioning, configuration management, and so on. This, of course, is the whole point of the integrated feedback-driven ecosystem approach.

However, if you set the CA portfolio against the backdrop of the typical delivery life-cycle, you get a pretty good picture of how a properly integrated framework operates in practice (Figure 5).

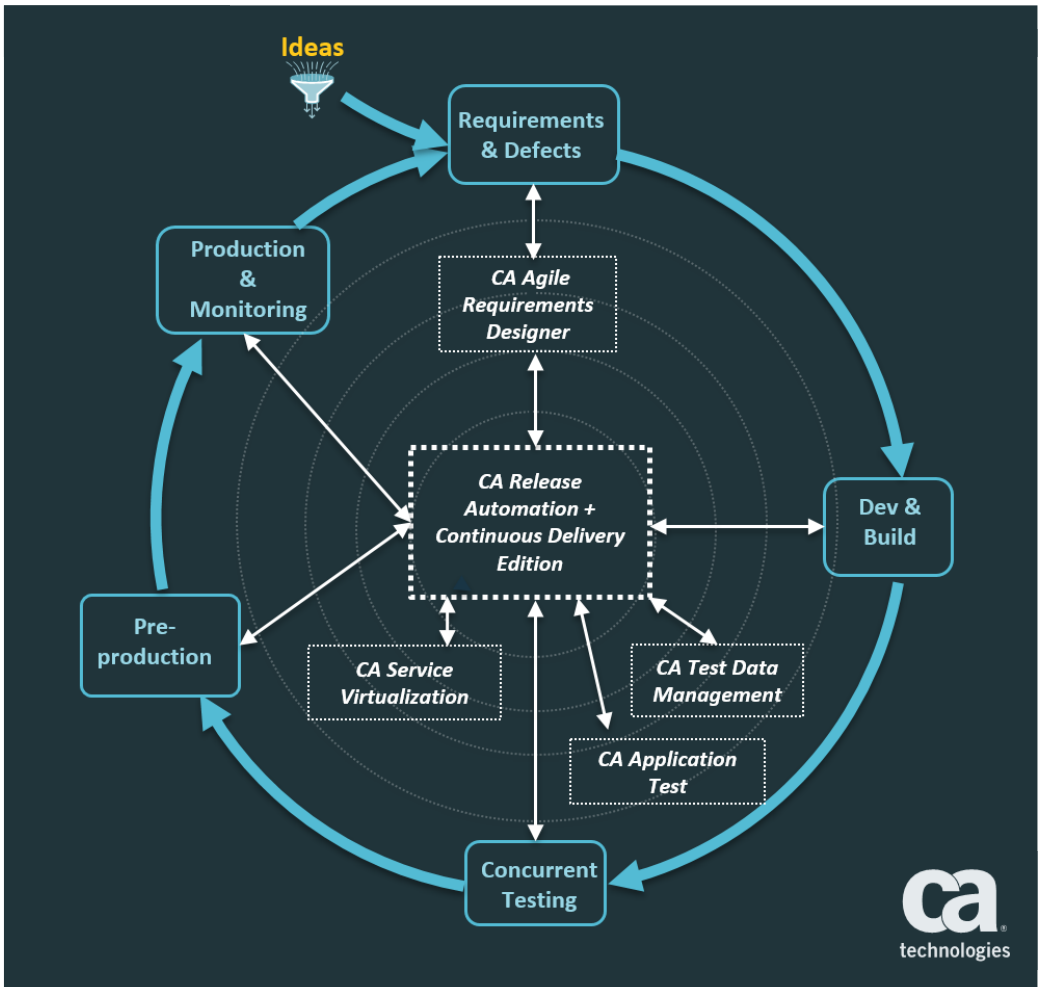


Figure 5

The CA continuous delivery framework in action, illustrating key integrations and activity flows

As you get into specifying your own framework, or reviewing your current DevOps environment, a common question is where you should aim to standardise versus allowing practitioner freedom. At one end of the spectrum, it often makes sense for teams or even individual developers to use the languages and integrated development environments (IDEs) of their choice in line with preferences or the job at hand. If one project team chooses Jira over CA Agile Central (or vice versa), then you may also be comfortable with that. At the other extreme, it is typically advisable to standardise on the solution you use for release automation as this represents a strategic point of management and control. Consistency here will help with visibility, repeatability, predictability and even compliance, and if you are managing a large number of interdependent application pipelines, a single enterprise-class release automation system is likely to be a key requirement for DevOps scalability.

Standardising on an enterprise class release automation system is likely to be a key requirement for DevOps scalability.

Making the business case

Continuous delivery can enhance both the top and bottom lines

No amount of DevOps-driven cultural change and reorganisation is a substitute for the solid tooling and orchestration capability required.

Continuous delivery is absolutely dependent on robust automation from beginning to end across the entire pipeline of activity, so you need a comprehensive set of enterprise class tools that work together seamlessly. No amount of DevOps-driven cultural change and reorganisation is a substitute for the solid tooling and orchestration capability required. In the vast majority of cases, implementing continuous delivery will therefore require investment. This in turn means being able to make a business case.

Given that every organisation and the point from which they are starting is different, it's impossible to come up with a prescriptive business case model or template. However, we can identify the main elements, around which you can put your own numbers:

Time to business value: With so many business initiatives now dependent on software, extended delivery times directly translate to opportunity cost. If ideas can be delivered into the hands of customers and users a week, a month or maybe even many months earlier, the impact on metrics such as customer acquisition, market share and revenue can be very compelling. The payback from earlier delivery is amplified in a particularly competitive and fast-moving market.

Beyond enabling the business more effectively, organisations that have moved from traditional methods to continuous delivery invariably see a significant boost to IT productivity.

Continuous optimisation: The ability to take an idea through the whole delivery process rapidly, safely and efficiently means customer-facing solutions can be continually optimised, especially with the right monitoring and feedback mechanisms in place. Sometimes a relatively minor feature, or even a tweak to an existing one, can increase cross-sell and up-sell performance with a direct impact on customer profitability. In a customer service context, the business impact is more likely to relate to reduced churn, enhancing profitability at a higher level.

Support of 'fail fast': Continuous delivery allows an experimental approach to be taken to product development, marketing campaigns and other initiatives. When getting first-cut capability into the hands of users is so fast and efficient, and rapid rollback can be achieved when necessary, you can prove or disprove concepts very quickly. Knowing that you don't have to strive for perfection on every occasion can be very liberating. Ideas flow more freely and more time is ultimately spent on concepts that work rather than those you just hope will work.

IT overhead reduction: Organisations that have moved from traditional methods to continuous delivery invariably see a significant boost to IT productivity. As a simple example, huge amounts of time can be freed up by automating and optimising test design and execution, both directly and as a result of lower remediation costs. Automating provisioning to avoid having to stand up, refresh and tear down environments manually can have similar benefits. Savings in these and other areas allow resources to be allocated to more value creating activities.

Raising expectations is part of securing the investment that will allow 10x or 20x gains to be delivered.

While sometimes difficult to achieve, educating business stakeholders on the first three categories of benefit is always going to be more effective. If you have any experience of managing budgets in an IT context, you will know that making a case based on internal efficiencies within your own team is always a difficult sell. Perhaps one of the biggest challenges is having the courage to raise expectations, but unless you do this, the kind of '10x' or '20x' gains that many talk about in this space are likely to remain elusive.

Final thoughts

Zooming out to the bigger picture

Transformation to DevOps and the implementation of continuous delivery go hand in hand. Automating legacy processes only gets you so far.

DevOps is clearly transformative at a cultural and organisation level, and continuous delivery is key to transforming the technical aspects of software delivery within this. The two, of course, go hand-in-hand. Trying to implement continuous delivery by automating processes based around legacy organisation structures and historical lines of demarcation will only get you so far. There is a need to stand back and rethink the way your delivery pipelines work, and challenge every point at which manual intervention takes place, custom scripts are relied on and handoffs occur. These are the typical indicators of a need for process transformation.

We should also mention that while our discussion in this paper has largely revolved around tooling and what we might think of as continuous delivery architecture, investments here need to be accompanied by the relevant practices. Whether it's making sure that developers commit their code into the mainline on a daily basis, or ensuring that policies on the generation and use of test data are adhered to, method and discipline are key to achieving success.

Continuous delivery can enable a step change in IT performance and bring everyone in development and operations closer to the business.

Finally, there's a whole discussion that we haven't got into around what happens in the production environment once a release has gone live. With so much emphasis on customer experience, basics such as performance and availability of applications cannot be forgotten. All that effort producing a great set of functionality and a compelling user interface counts for nothing if the service breaks down or runs painfully slow. There is then the whole area of monitoring and feedback, whether at a functional or performance level. But these are topics for another day and another paper.

In the meantime, we hope our discussion has illustrated how continuous delivery can enable a step change in IT performance and bring everyone in development and operations closer to the business. It only remains for us to wish you success with all of your efforts in this area.

Further reading

The following related reports are available from the Freeform Dynamics website:

Assembling the DevOps Jigsaw
Do you have all the right pieces in place?
http://www.freeformdynamics.com/fullarticle.asp?aid=1868

Exploiting the Software Advantage
Lessons from Digital Disrupters
http://www.freeformdynamics.com/fullarticle.asp?aid=1867

APIs and the Digital Enterprise
From operational efficiency to digital disruption
http://www.freeformdynamics.com/fullarticle.asp?aid=1870

Appendix A: Research overview

The study providing the data included in this paper was designed, executed and interpreted by Freeform Dynamics Ltd in collaboration with CA Technologies. Data was gathered from 401 respondents working in medium and large organisations via an online survey completed in May 2016. Countries represented included Germany, Italy, UK, France, Spain, Switzerland, Sweden, Netherlands and Denmark, and the sample covered a cross-section of industries. 28% of respondents were overall heads of IT, 21% were departmental managers, and a mix of middle ranking team leaders, DevOps specialists, architects and other practitioners made up the remainder.

For analysis purposes, the study sample was divided into three performance groups based on their responses to a number of outcome related questions (Figure 6).

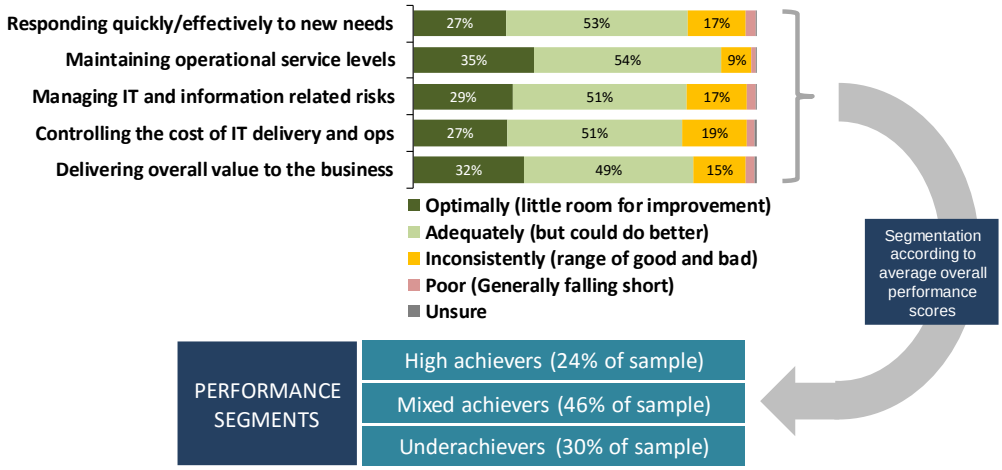


Figure 6

Segmentation based on responses to the question "How well would you say IT is delivering at the moment against the following?"

Note on the research methodology

The survey was conducted online and respondents 'self-selected' into the study. We must therefore be aware of possible sample bias towards more advanced respondents who are generally more enthusiastic and more likely to respond to a research call to action. This does not affect the commentary or conclusions contained in this paper, but should be borne in mind when considering the data in another context. Questions relating to this or any other aspect of the research should be directed to info@freeformdynamics.com.

About Freeform Dynamics

Freeform Dynamics is an IT industry analyst firm. Through our research and insights, we aim to help busy IT and business professionals get up to speed on the latest technology developments, and make better-informed investment decisions.

For more information, and access to our library of free research, please visit www.freeformdynamics.com.

About CA Technologies

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. Learn more at www.ca.com.

Terms of Use