TQ Agile & DevOps

by Accenture

Resume Course

Bookmark Add to Channel Download Course Schedule Reminder

Table of contents Description Transcript Exercise files Discussion Learning

TQ Agile & DevOps Preshow

Transcript links

TQ Agile & DevOps Preshow

Agile and DevOps are the focus of this TQ topic. And while they're not technologies in their own right, like many of the other TQ topics that we've done, Agile and DevOps are extremely important for technology because they provide the framework. They provide a mindset and a faster, more innovative way to implement technology and bring enormous value to Accenture's clients. Agile and DevOps are a core part of our daily lives and how we help ourselves and our clients get their new and improved products and services out to their markets much faster. Now I'm sure you've heard of both of these terms before. The markets may refer to Agile and DevOps as methodologies. Now at Accenture, we believe that Agile and DevOps transcend that. They're a complete culture and mindset. Agile is a set of core principles and frameworks in which customers, product owners, designers, developers, and others all work together in an ongoing and iterative way to push out a minimum viable product or MVP. This'll is an initial product that they're going to try in production and then continually iterate on or improve on that work product as time moves on. In short, Agile is a way of working. Now Agile differs from the traditional waterfall methodology that many of us are familiar with, and it does this in a number of ways. In a waterfall approach, the product is developed in a more linear fashion. That is, you define your requirements, you design the product, you develop the product, you test it, and then you actually release it to production. There's no real room in a waterfall approach for changes along the way. Now once those requirements are defined, they're locked in. And if you do have changes, you need to start the entire process all over again from the

beginning. And it costs incrementally more money the further along you are in the work product. Waterfall is regimented, and it's a much more disciplined framework. Now, this actually doesn't imply that Agile is undisciplined. It's actually very rigorous in its own right as it assesses quality and fit for purpose in every sprint of iteration, and that's super important. Don't get me wrong. We do a lot of Agile work for Accenture's clients. However, we also still do waterfall, and that's okay. It's all about helping our clients choose the right approach for them and for their business challenges. Now DevOps is a newer concept in the software development world, but it started almost 20 years ago, and it literally means development and operations together. Now there's a lot of variations of DevOps out there. You have DevSecOps, you've got DevXOps, and many others. But for simplification purposes, we're going to be calling this just DevOps. In a traditional software development environment, you would have your coders and your developers build a work product and then actually deploy that code into production where operations team would then make sure the work product runs smoothly and they fix any bugs found in production. Now you might imagine that when bugs or issues are found in the code that this leads to a lot of finger-pointing. Well, I'm not imagining that. A lot of us see that. And blaming where those problems came from and who should be responsible for fixing it, well, that was one of the key challenges of waterfall-type approaches. Now DevOps is actually an Agile way and a form of working where the development and operations teams actually work collaboratively for the best product before and during production. And it borrows some of the same core principles of Agile, a cohesive team with a shared objective. They're embracing failure along the way. That's a really interesting thing to consider that it's okay to fail because you're going to put guardrails around it so that it doesn't cascade into bigger problems. Now iterative development and experimentation without regret, these are also key principles of working in a DevOps manner. Now in that sense, it's more of an organization construct for how two or more companies or two or more groups should actually collaborate. So why do we feel that it's so important for us to talk about Agile and DevOps in TQ? Well, it is proven that using Agile and DevOp approaches have really sped up the time to market for many of our clients. In fact, some of them see almost a 10x improvement over traditional waterfall methods. They also see better product quality by as much as 80% or up to 50% more productivity. And honestly, it actually increases team morale because everybody's closer to seeing the finish line all the time. Agile and DevOps started to get more popular when businesses saw companies like Apple and Google and Netflix pushing new products and services at a torrid pace out to the market. Now the larger brick-and-mortar companies out there, many of whom are our clients, saw their industries getting disrupted. And they needed to figure out a way to adopt and do it really quickly. And this led to an increase of Agile practices around the world. Now Accenture not only uses Agile and DevOps for ourselves, but we also have a substantial practice of consulting and advising our clients on how to adopt it in their own organizations. In the spirit of one Accenture, we actually have multiple practices that focus on Agile for our clients. For example, our SolutionsIQ practice focuses on advising our clients on business agility, while our talent and organization groups, strategy and tech consulting, and other organizations focus also on advising and implementing Agile solutions with our clients. And then, of course, in technology and our operations businesses, we focus on delivery agility. Today, we have over 75,000 Agile practitioners and more than 13,000 DevOps experts within Accenture around the world. And we have more than 400 active

Agile-consulting engagements at any one time plus more than 250 clients that are being supported by our DevOps innovation centers. So we're really doing a lot in this space, but there's more ground to cover still. And Agile and DevOps are a major part of Accenture, and it's all around us. Even if you're not a technical person, I'm sure that you're exposed to or using Agile in some way. If not, I really challenge you to learn more about Agile and DevOps and see if there's a way to bring these principles to your daily work to improve the way that you and your business actually operates. Now you should know what these concepts are, and you should also be able to have a conversation about them and understand what their value is. So for everyone out there, to grow your TQ, definitely pick up Agile and DevOps. It's going to help you grow your impact. Happy learning.

Agile & DevOps: Executive Briefing

Introduction

In any organization with active software development projects, there are two words you'll often hear regardless of what specific platforms or technologies might be in use. Those words are Agile and DevOps, and these are two different subjects. But more and more, they become part of the same conversation. Any time we talk about best practices or approaches to plan and deliver successful software, you'll hear the terms Agile and DevOps. But what did these actually mean? What's the difference between them, and why are they considered complementary? But most of all, how do they help? What value do these two ideas bring to current software development? Well, let's find out. Welcome to the Executive Briefing on Agile and DevOps.

The Waterfall Problem

Agile and DevOps are easier to understand once you realize they both try to fix very specific problems, problems we encountered again and again using the software planning and development methods that were typical from, let's say, the 1950s all the way through the 1990s. And, okay, I'm not saying that every organization would plan and build software exactly the same way. But still, for several decades, if you were involved in any large-scale software projects, you would expect it to be planned and managed using a formalized, highly structured approach with a sequential set of different phases or stages, something we now call a waterfall model or a waterfall methodology. The term waterfall wasn't the official name of anything. It's now what we call the various project planning approaches, whereas each phase of the project was completed, the results coming from that phase then flow down to the next phase. Generally, the waterfall methods for software projects had five or six separate phases. First, often considered most important was requirements gathering where the team would perform research, interview customers, and do whatever else was needed to attempt to fully and completely

understand every single aspect of what the software would ever need to do and document all of it in excruciating detail. And we're not talking about days or weeks of work. The requirements phase would often take months to complete. If you're wondering whether you get some proof of concepts or demo applications from this, no, nobody was writing any code. Nobody was programming anything yet. What you get from the requirements phase was basically a thick binder with hundreds of pages of documentation. This would be the source of authority to drive every decision from this point on. The requirements phase was then complete. Then we move on to the second phase, software design or software architecture, to now define the technical aspects of exactly how this software was going to be built. The output of this design phase was also more documentation, which then flowed down to the next phase, development or implementation, where the programmers would actually write the software. So we were several months in, sometimes years, and the project hasn't really even started to be coded yet. After development was finished, it would flow down to the fourth phase, testing, usually performed by a separate team. If they found issues or bugs, someone had to decide, Should that bug be fixed? Could it be fixed without messing up the project timeline? Or could it wait and be fixed in the next version, whenever that was going to be? And after that was done, we'd reach the deployment or operational phase. And deploying a big piece of software to a new environment almost always runs into unexpected problems. But, remember, by the time the team first started collecting requirements and the time the software was finally deployed, years had often passed. It's entirely possible that the original requirements had changed or they weren't even properly understood. The customer, who sometimes hasn't even been talked to since the very first phase, may take one look at the software and say, That's not actually what we wanted, or that's what we wanted two years ago. It's not going to work now. We needed feature X, and we needed it six months ago. And if there's one thing a waterfall approach is not good at, it's going backwards up the waterfall to deal with new requirements or changes in the business landscape, or even just things that we learned and could have been better along the way. A waterfall approach has its place. If you're building a suspension bridge, there are reality-based reasons to have specific or independent planning, architecture, construction, operations phases where you want to micromanage certain parts of the project. But for software, it's too constrained. It's not responsive enough. It's not flexible enough. And it's not fast enough. And most important, waterfall doesn't support what software development is naturally good at, evolving and changing, being updated, optimized, improved, extended. So developers and businesses started looking for a better way to build and deploy software projects and being more flexible, faster, and more responsive, a more Agile approach.

Introducing Agile Development

I'm not suggesting we all happily used waterfall planning methodologies for 40 or 50 years before realizing there was even a problem. No! People recognized issues with that heavyweight micromanaged approach very early on. And over the years, there have been many suggested methods for planning the software development process. But particularly in the 1990s, alternative approaches started to become more popular, methodologies with names like The Unified Process and Scrum and XP, or Extreme Programming. And just to be clear, these

aren't technologies. They're just ideas, guidelines, suggestions for a structured approach to plan a software project. But while these different approaches, these methodologies weren't identical, there were a lot of people trying to fix the problems of waterfall in a similar way. So in 2001, 17 of these people, including folks working on other methodologies like Scrum and XP and well-known, well-respected developers and authors working on software best practices, got together and wrote the Manifesto for Agile Software Development. Manifesto sounds intimidating, like we're about to get another big binder of documentation, but it's not. You can write the Agile manifesto on one page. It's a set of values and principles. And, okay, values and principles always sounds a bit vague and a little hand-wavy and fluffy. But it's specific. Four values, 12 principles written in plain, non-technical language. And if you're wondering what's the difference between a value and a principal here, values come first. They lead to the principles. Values are that high-level, What do we value? What do we think is most important about software development? Then the principles become the more practical, Okay, if that's what we value, how are we going to make it happen? And, again, these values are all in plain, non-technical language. Let's take a look.

The Agile Values

As we go over these values, remember, the Agile manifesto did not come out of nowhere. This was a course correction. It's aware of and trying to fix the problems that happened again and again with the waterfall project planning. So here each of the four Agile values is written as a comparison, a contrast between two things. We've prioritized this above that. For example, not just we value working software, but we value working software over comprehensive documentation. So this value recognizes that all waterfall projects often encouraged an over-focus on exhaustive documentation where more effort, more time, more priority, more attention was placed on that instead of on the actual software development. Quick sidebar. Some critics of the Agile approach have misrepresented this to say Agile doesn't do any documentation. It's not true. Documentation is great. You should have it whenever and wherever you need it, but never lose track that the entire point of doing this is working software, not just a shelf of impeccable binders. Let's take the next one. We value individuals and interactions over process and tools. Software development is not a predictable, numbers-based, pure resource management task. People did try to treat it that way. We have five developers on one team. They say it'll take a month to implement a new feature. Well, let's just get an agency to provide 15 temps, and we'll have it all done by Friday. It doesn't work like that. There's technical knowledge, institutional knowledge, technical ability, productivity, team dynamics, multiple skill sets required, and a host of other things to take into account. And notice the very intentional word choice here. This doesn't say we value people, which would be cliche. Here, it says, we value individuals. Agile asks us to not just focus on anonymous roles to fill or jobs to be done, but to always bring our attention to the actual individuals on the team, their personalities, their whole selves. Who, exactly, who is part of this? What's that person's technical skill set? What's their business knowledge? What are their strengths? How well do they work with others? Very importantly, how do all these different individuals communicate and collaborate? But, again, this value is not saying processes and tools aren't important. They are! But your

individuals and how they collaborate is more important to the success of a project. Next, Agile development values customer collaboration over contract negotiation. We're not looking to find the minimum number of line items we can get a customer to agree on, then refuse any other requests because that wasn't in the original contract. These days, successful software development must be a partner to the business, not just a service to it or an enabler of it, and certainly not in competition or conflict with it. But, again, this is not saying that contracts are unimportant. They are, but as an agreement and an understanding, not as a club to beat each other with. The higher value is genuine and ongoing collaboration with the customer. And, finally, Agile values responding to change over following a plan. We bake in the idea that change will happen. There will be new demands, new competition, new regulation, new devices to support, new understandings. New, new, new. Well, we don't know what might happen. That's the point. And we encourage our own abilities to be responsive and flexible. But, okay, these are all good. But there's nothing here that says, How do we actually do any of this? And that's where the Agile principles can help.

The Agile Principles

We're not going to go through every single one of the 12 principles, but let's take this first one. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. In this one, deliver working software frequently, from a couple of weeks to a couple of months, with the preference to be shorter timescale. So even just these two alone is a huge perspective shift from waterfall, where we might talk to the customer early on and then disappear for a couple of years. And to support these principles requires us to take an iterative and incremental approach. Instead of massive, long phases of waterfall, we assume that here we'll have multiple cycles where each cycle involves aspects of requirements, planning, implementing, testing, and deploying, and then start the cycle again, delivering something valuable, something the customer can try and test and give feedback on every time. Next, agile tells us to embrace the idea that requirements are never fixed in stone and always changing and evolving. There are several agile principles around the idea of communication between the business and the developers, and communication within the team itself, trusting the team, allowing the team to self-organize. And what these principles lead to, even when it's not explicitly stated here, are some typical practices. For example, in waterfall development, it wasn't unusual to have teams of 40 or 50 or even 100 people. Agile teams are usually small, typically 10 or so people. Sidebar, we suggest a two-pizza team, meaning If you need more than two pizzas to feed the team, it's likely the team is too big for efficient communication. Small teams are able to collaborate more easily and react more quickly. Teams may be colocated or collaborate virtually. And there are other terms you might not see in this list of agile principles but have now become very associated with agile. You'll often hear agile teams talk about sprints or iterations, which are defined periods of time, aka, a timebox, often only 2 to 3 weeks, where the team builds and tests and releases new software. In a typical sprint, the team will take a look at their to-do list, called a backlog. They make a quick estimate of which items they can get done during the sprint and then start working. It's all about breaking up the workload into small chunks and working on and delivering one chunk at a time. And agile emphasizes technical excellence.

Quality becomes the continuous responsibility of the team itself, unlike in waterfall, where was passed to the testing team to make that judgment. This focus on quality, shorter delivery cycles, and faster feedback on working software has led to the almost ubiquitous practice in agile teams of unit testing. A unit test is a small automated test that focuses on one single behavior or operation that the software is supposed to perform. And the developers continually write and save new unit tests as they're writing the software itself. Testing that becomes part of development, not something done afterwards. Developers can work on a new feature and with a click of a button, quickly run a set of automated tests to see if it's working as intended and make sure it didn't break anything else. This unit testing reduces the time to release and increases the quality by replacing larger, more time-consuming, full-scale application testing that waterfall projects relied on. However, while the agile principles talk about continuously delivering software to the customer, they don't talk about how we can reliably and repeatedly do that, and that's where DevOps comes in.

Sidebar: Is Agile a Methodology?

A quick sidebar moment. It's not unusual to hear people talk about the agile methodology, but this word methodology is problematic and a lot of folks in the agile world try and avoid it. First, the manifesto is intentionally not prescriptive. It doesn't pretend to be a methodology. It isn't giving you this formalized project management template that you're supposed to follow. That's kind of the point. One of the values here is to stop thinking that if you just find some process in some external methodology and then follow it to the letter, it'll take care of everything. We're trying to get away from that approach. But structures and processes are still important. They can provide consistency, clarity, clarity around expectations, and promote transparency. Agile frameworks like Scrum, XP, or Crystal all support the values and principles in the manifesto, but then add some specific guidelines and practices around things like planning, facilitating meanings, managing communication, distributing workloads, or keeping track of feature requests. You'll notice that these are called frameworks. Even they avoid the term methodology because it suggests the right focus. We're not trying to find a methodology to follow. We're trying to find a framework that is flexible and can support us in building something. End sidebar. Back to DevOps.

Introducing DevOps

DevOps is a mix of the words development and operations, and like an agile, it aims to be fast, responsive, and focused on the customer. And while you won't see the term DevOps appear in the Agile Manifesto, they complement each other very well. They go together like milk and cookies. Instead of deployment and operations being stuck at that final tail end of the waterfall process, it becomes incorporated into the full delivery lifecycle. Everyone working together to deliver value to the business, and DevOps has a strong focus on automating as much as possible. For example, developers may, as part of creating their software, have to create a description of what their software needs to run, what kind of operating system it needs, what components it uses, what frameworks it relies on, and so on. So let's say a developer finishes a bit of code and wants to test it. They check

the code into the source code control system. An automated process, often created by the organization's DevOps team, would automatically begin. That process might start by reading the description of the software's environmental requirements and then actually create that environment. The process would then deploy the new software into that environment and kick off the automated unit tests. If the unit tests pass, then the software is deemed acceptable and is ready for deployment. If a test fails, the developer is notified, and they go back to programming. Some organizations will take successful software and run larger-scale integration tests for other validations. But at the end of the day, the software is ready to deploy, and that deployment can also be automated. For example, in a cloud-based environment, the new application code might be launched into containers or virtual machines that are emulating whatever hardware or platform we want to use. The containers or virtual machines running the old version of the application would then be deleted, and users would find themselves using the new application right away, hopefully without realizing anything had even happened. This entire process is often referred to as a pipeline. Depending on the specifics of an organization's pipeline, you might hear terms like continuous integration or continuous delivery or the CI/CD pipeline. You've probably encountered this even more than you realize. Something as simple is your smartphone automatically updating one of your apps is agile in DevOps practices in action. A team has completed a sprint. It's gone through things like automated unit testing and continuous delivery to automatically push out the update, and the result is you have an enhanced app with new features and bug fixes and hopefully with very little or even no interruption to you as the user. So that's it in a nutshell. At least from a software development perspective, Agile is about producing small improvements to an application in a continuous series of short sprints. DevOps helps developers automate the testing of their code, and when all the tests pass, DevOps can also help automate a pain-free deployment.

Implementing Agile and DevOps

Both agile and DevOps are really philosophies. They're a set of principles and values that encourage a way of thinking, a mindset. There's no one correct way to do them. You'll find frameworks such as Scrum or extreme programming that bring specific approaches and practices to the table, but they're agile because their practices support the values and principles. Agile and DevOps bring new ways of working into an organization. Most organizations start by shifting a small number of projects to agile and DevOps. Doing so lets them gain some experience and evolve their practices and processes based upon what they learn. Agile and DevOps can be applied to customer-facing software applications, such as mobile apps and websites. They can also be applied to internal-only software, such as custom line-of-business applications. Anywhere that you have software that can benefit from smaller, more rapid, more frequent release cycles, that's where agile and DevOps are valuable. And here's the best part. Once organizations got a taste of agile for their software development, they started wanting those benefits across the business and elsewhere in the organization. They were being agile with a capital A, but they wanted to be more broadly agile with a lowercase a. Adopting agile practices and leveraging agile values and principles in other areas is often called enterprise agile and leads to business agility.

The reason agile principles can be applied so broadly is because that's all agile really is. It's a set of principles and values. They may have their roots in software development, but today we recognize that agility has value across the entire organization, even on internal projects. A finance team might roll out a new expense reporting system, monitor it for a few weeks, and then roll out the changes based upon what they've observed. Or a strategy team might adopt a new pricing model for their company's services, gauge the market reaction, and then adjust. A marketing team might forgo that $1 million campaign of the old days and instead focus on a series of smaller campaigns that lets them sharpen their aim. The point is the agile mindset has helped organizations become more adaptive, how to quickly roll out something small, see how it does, and then move onto the next set of improvements. Instead of having to make massive, expensive, multiyear investments and then pivoting to react to the market, now companies can continually test ideas, features, and react more quickly to identify their best path to success. They become, well, more agile.

Agile and DevOps in the Enterprise

However, there are some issues about agile that can make it difficult for some companies to adopt. Agile is a very intentional way of working, and it applies to all levels of the organization. You might often think about agile practices being applied at the team level, but in order for that to work, their managers have to understand, agile and support it. And to do that, the organization's executives have to buy into agile as well. That executive support is important because moving to use agile principles requires a compelling vision. Organizational leaders need to clearly communicate why they want to make this change and what they expect from it. They need to relentlessly share that vision and help everyone understand this is where we're going, and this is why. It's also important to understand that change can be hard, and for many organizations, agile is a big change. Agile and DevOps aren't an all or nothing proposition. Some parts of an organization may use them, while others don't. Agile and DevOps are really about mindset. You can leverage them anywhere they make sense, such as on specific software projects or a specific marketing campaign. You can also use other methodologies and practices in places where they make the most sense instead.

Tools and Environment Overview

Neither agile nor DevOps impose any kind of specific platform, technology, or tool. Agile software development uses many of the same tools that developers have used for years; integrated development environments such as Microsoft Visual Studio or Eclipse, source code control systems like Git, and, of course, whatever languages or frameworks they've always used. Unit testing is usually a part of agile in DevOps environments, and a huge variety of languages and tools exist for creating and managing those unit tests. Although not specifically tied to agile or DevOps, technologies like virtual machines and containers usually crop up in environments that are moving to an agile and DevOps approach. These technologies enable faster deployment of software for testing and to production. DevOps pipelines often involve one or more sets of tools designed to facilitate automation. These may include automated testing software, continuous integration, or delivery tooling, and so on. But

remember, simply having the right tools doesn't magically create agile or DevOps. These often require a great deal of education and thoughtfulness to successfully adopt. They'll often require buy-ins at the highest levels of the organization, and they may even require significant cultural changes, org chart realignment, and more. Tools and technologies can certainly support the use of these principles, but tools alone won't create the outcomes you're looking for. And remember, agile principles aren't limited to software development at all. You can find other tools that can support agile throughout the organization, like Kanban. Even good old whiteboards can be useful along with real time collaboration tools. It's all about sharing information within and across teams, aligning and coordinating efforts, and breaking those giant workloads into small, rapid sprints. So there you have it, agile and DevOps in a nutshell, designed to help organizations create, evolve, and deploy more quickly and efficiently. They have been changing the way we think about software development for years. Now they're starting to change the way we think about everything our organizations do, from finance and marketing to nearly any part of the business. And with all that history behind it, we're seeing more and more organizations start to explore their benefits. As markets and consumer needs keep evolving, agile organizations will be ready to respond rapidly and bring value to their customers. I hope you found this useful. Thanks for watching!

TQ Agile & DevOps Aftershow

Agile & DevOps Intro

-Everyone is talking about Agile these days, but do you know what Agile and its close companion DevOps really means? Well, if you're curious, you've come to the right place. And if you're thinking Agile and DevOps don't apply to you because you don't work in software development, well stay tuned. It's all coming up on this next edition of the TQ Aftershow. Hi, Adam. Thanks for hosting this edition of the Aftershow with me. -Hi, Sarah. I'm so happy to be here, in fact, I'm especially proud at how agile we're actually being in doing these after shows from home now. -Yeah, I see what you did there, Adam. That's a great segue because today, we're talking about Agile and DevOps. -Yeah, correct. Sarah, I'm actually glad to really talk about these topics, and especially now. I really want to clear up a couple of misconceptions about them. -So, misconceptions, like, what do you mean, Adam? -Well, first of all, a lot of people think that when they do anything in an agile way, that they're embracing Agile with a capital A. -Yeah, I understand what you mean. I've heard people use agile in lots of different ways, and it's not always clear what they really mean. -Yeah, exactly. And, you know, you're right, the word agile can actually mean a lot of different things, and we'll get to that a little bit more in a couple of minutes. The second misconception is that Agile and DevOps are actually methodologies. -Hmm, okay, and that's interesting because I definitely thought that they were both methodologies. -Actually, they're not methodologies, and later on my good friend Jeffson Dsouza, who's our local Agile professional community leader, is going to talk more about that. And the third misconception is that Agile and DevOps are only for software development. That's not true at all. -Well, I'm definitely interested in that, and I really want to hear where else they can apply because I thought

we were only talking about software development. -Yeah, you absolutely can use those things in any part of the business, and for this, we're actually going to have someone from Accenture HR, Allison Horn, talk about Agile and DevOps work that they're doing inside of our Talent organization. And last but not least is that Agile sacrifices quality for speed. That also is absolutely not true. -Sounds good, Adam. I definitely look forward to clearing up all the misconceptions today. And you've already introduced Jeffson and Allison, so thanks for doing that and making my job easier. But on today's show, we have several other guests that are going to join us, so let's meet those now. We're also going to be joined by John Rudd, Neville Poole, and Mirco Hering. John our global business Agility lead, Neville is our North America business Agility lead, and Mirco is our global DevOps practice lead, who I hear literally wrote the book on DevOps. So John, Neville, Mirco, Jeffson, and Allison, welcome to the show. Hopefully by now, all of you've had a chance to watch the executive briefing, and you know what Agile and DevOps are. So, we're really going to jump right in and talk a bit more about each of these in detail. So, Adam, let's start the conversation with one of the misconceptions you mentioned. You said that Agile and DevOps are not methodologies. So, tell us a bit more about that. -You're absolutely right, they're not methodologies. That question really relates to the elephant in the room here, and I want to address it before we go any further. All of our other TQ topics are really focused on a specific technology that businesses can implement to make their business get to market faster or perhaps better adopt to a changing world. Now Agile and DevOps, they're not technologies that we implement, instead, they're the ways that we implement technology. So, when we're talking about Agile and DevOps, and we're doing it together, it's because these concepts and mindsets are actually ways of working with these technologies to help bring enormous value to our clients and importantly, faster. Now, here's where we want to differentiate Agile and DevOps from methodologies. When you think about a methodology, it's a lot of processes, and tools, and rules. Don't get me wrong, Agile and DevOps, they have those things too, but they're much more than that. Agile is a set of values and principles, a mindset, if you will. In fact, people in businesses need to think differently when they adopt Agile. It's actually a completely new way of working. It's a dramatic shift away from the mindset of gathering requirements, doing design, writing code, testing, and then deploying, and then maybe getting some customer feedback afterwards. In fact, it kind of turns the whole process a little inside out, starting with the end in mind, collaborating with customers upfront and throughout the entire process, constantly iterating on the requirements, the design, the feedback through the entire development process, and being comfortable with something that we call an MVP, or minimal viable product, before getting the final polished end solution. DevOps really is a lot of the same. The goal of DevOps is to deliver services faster and at a higher quality with security built right in from the very beginning. To achieve this, it encompasses a lean engineering culture, technical practices, and it emphasizes collaboration, communication, productivity, automation, agility, and high-quality shift left in the development process. -Wow, Adam, that was a lot, but something you said there in that last statement caught my attention. I've heard this phrase shift left a lot. I'm guessing it doesn't have to do anything with driving a car, but what does it really mean? -Yeah, it definitely doesn't have anything to do with driving a car, although ironically I can actually shift a car with my left hand now, since I've been living in Singapore for the last 4 years, and they drive on the wrong side of the road, but I digress, in the context that I'm talking about here, the

shift left that I'm referring to is about shifting key operations tasks to earlier in the development process. You know, that way, you catch problems much earlier, and you're going to avoid a lot of quality issues and rework. And I'd actually like, at this stage, to have Jeffson expand more on this concept of Agile and what some of the key guiding principles are. Jeffson.

Agile Principles & Values

Sure, Adam. Hello, everyone. And let me explain to you the concepts of Agile and also how they relate back to our principles at Accenture. Now, as Adam rightly said earlier, Agile is not just a methodology. It's a way of working where development, operations, and business, they work together as one team. They collaborate, provide feedback, and continuously look to improve. The processes and tools are just about helping the team to be more transparent and bring collaboration in between them. In the Agile Manifesto, one of the Agile values is about people and interactions over processes and tools. However, in the commercial context in which Accenture works, we place a lot of importance on process, scope management, and working to agreed outcomes as a part of our contracts. That's the first principle. The second one is while we're working with Agile, we're working with the clients throughout the entire process. I can't emphasize that more. If you're working on an Agile project, it really means that you need to really be working with the clients. And just to reiterate that point again, let's not get caught in the difference between Agile with a capital A and agile with a small a. Working very closely with our clients is a fundamental principle of how we work at Accenture with Agile projects. -Thanks, Jeffson. I love that. And some of the themes that I heard you say just now remind me of some of the past aftershows we've done when we've talked about block chain and cloud and data. All of these things are around technology, but they really require just a different way of working, which I think is so interesting. And the other theme that always comes up when we do these aftershows is around value and how Accenture is helping our clients use these technologies and these ways of working to really get more value. So let's talk a little bit more about that. John, maybe you can come into the conversation and tell me a little bit about the business value that companies get when they adapt Agile and how we at Accenture are helping them do that. -Yeah, sure. So, it's interesting, this space has evolved quite a bit as it relates to why organizations turn to it initially. And you've got to remember, this is over 20 years ago that people came together and created a manifesto, which is this group of principles. And the principles themselves are quite simple, but what the dawn of this was really the frustration that was going on in the industry and the development industry at the time. And the frustration was around the fact that organizations were making these big, huge business cases around big, huge projects. And then they were, you know, the business would put together a big stack of requirements and they'd slide that over to delivery, and then delivery over the course of three years would build this out, and inevitably, you know, despite the fact that these things were typically 50% or more over budget, 50% or more over time, inevitably, the end result was not meeting the satisfaction of the business. So the IT guys and gals were getting kind of blamed for this, and it was very frustrating. So, initially, it was really quite simple, and that was to focus on kind of a project management process that would ensure that there was going to be quality, ensure that it was going to be done

in the way in which there was going to be confirmation that the business was accepting. So when you actually take a look at some of the core different practices around Agile, it's around making sure that there's something that's deliverable, that it's fairly small, and that the business is nodding their heads saying, yeah, that's about right. So that was kind of initially what it was, and it was a breakthrough moment. The problem with it at the time was in the early days, everyone kind of felt well, that's fine if you've got one or two teams, that's fine if you're working on a green field, but we're a large Global 2000, Fortune 500 companies, we have big, complicated legacy systems with multiple dependencies. So, as this kind of evolved, the value started to move north, but only north when we started to get more and more sophisticated around DevOps and some of the automation practices, so that there actually could be larger deliveries. And then there were scaling systems that came into play, say, and others, that allowed across an organization the ability to still have some level of consistent delivery on an ongoing basis and in these very complicated areas. So, when we kind of got to this level, what happened is the whole arrangement between business and IT changed. Where there had not been a lot of faith on the business side, that if they asked for something it would be done right, and there was a lot of pointing fingers, and those kinds of things, but this consistency, this drumbeat of delivery that started to come out built up the confidence in the ability now for business and IT to start to work more closely together. And the reason I kind of tell the story is as this was going on out in the marketplace, what was going on is that we saw technology platforms on top of technology platforms that were leading to new product innovation that people hadn't seen before, you know, the tenure on the Fortune 500 companies was shortening and shortening as new companies were coming up, so there was all this disruption that was going on. And that's when organizations started to say, hey, the fact that we've actually had the ability to build something small, the fact that there's this disruption, and the fact that we can't plan three years in advance, we needed to get into a little bit more of a test concept. We need to be experimental. And this really fit very well into these Agile values, this concept of being able to build something small, test it, test it with the marketplace, provide feedback loops on an ongoing basis, and then be able to alter, change, do other things, there then was a development of persistent teams with persistent levels of quality, getting away from a paradigm which was project into one that was product and flow-based, and all of these components came together so that you could get this idea of going from concept to cache shortened. And so when you start to think of things like cycle time becomes really the key because being able to shorten that feedback loop gives you that much of an advantage over your client in terms of what's going on in the marketplace, and then you can actually take a right or a left-hand turn as is necessary. So what went from how do we get through kind of these big projects and make more sense out of them has really led to organizations now having the capability to be responsive, to be adaptive in how they're doing things, and then this builds a core competency around continuous innovation. So, it's now a centerpiece for a lot of what organizations do, and I would say that a lot of our clients that we work with would identify that this capability is absolutely essential, not only for them to hone and to thrive, but for them to survive as they go forward. And to kind of make this real, I mean, Neville, we work with clients all the time that are doing these things. I know you've got some great examples, but maybe you could share with the group here some thoughts with maybe one of our clients in terms of how they are leaning in and taking advantage of this type of value opportunity.

Agile Client Examples

Yeah, yeah, sure. Absolutely, John. So one example, back in the beginning of this year, back in January, we kicked off an engagement with a global food retailer that was feeling a lot of the pain that John just described. They were behind in their overall retail industry. They needed to operate faster. Their quality was suffering. And they really wanted to embrace this culture of experimentation. But they didn't really know how. So we started with the discovery, and we found a couple of issues that were leading to their struggles. The first one was they were working very much in silos all across the organization. So there was no real collaboration, just handoff after handoff after handoff. So, two, they lacked product management focus. They were all about output and quantity versus really thinking about, What business outcomes do we want to deliver to our customers with very high quality? The third thing was they had just such a rigid process, and the governance structure really made it almost impossible to accommodate any type of change. And the last thing was fear of failure. No one was really willing to go out on a limb. It was all or nothing. It had to be right. And so there was this fear of not getting it right the first time. So after understanding those four primary challenges, we mobilized the team of about three Agile process coaches and a technical coach to really begin this Agile transformation but in a small scale, two programs, and those two programs really focused on getting right size teams. How do we bring a new team construct together, business and IT operations, together on one team to really understand what the problems were that the business was trying to solve and solution those and deliver those, design those, test those, all together as a team? The second thing we wanted to do was really help them kind of grapple with the fundamentals of those principles that John just described. How do we create transparency and collaboration across this small group of people? How do we get predictable? How do we start to really perform as a high-performing team? And then we wanted to really get a baseline from a metrics perspective. Okay, how long is it taking to understand what it is that we want to deliver, and how long does it take us to deliver it? So we really wanted to get those fundamental principles embedded in the team, that mindset shift from handoffs to collaborative design, collaborative build, and collaborative testing so that they can get incremental value out the door quicker and get feedback from the customers. So after a real short period of time, about three months, they recorded their highest employee satisfaction scores to date and have really laid the groundwork for building predictable, high-quality teams, and now going into scaling. -Yeah, I love that example because it's huge, as you said, I mean, changing a company's culture is no small act, and you did it and showed them in a way that they could see the results and see the impact. I mean, engagement scores raising. That was awesome. And then they could see the results and move on from there. It's a great example. Jeffson, maybe you can just sum up in your mind what are some of the typical values, what are some of the ways that we see our clients really get the value out of Agile DevOps? -Thanks Sarah. Let me share another example of one of our NA based insurer customer who started Waterfall, then moved to Scrum, and then moved to SAFe, since they are a large team spread across the globe and were using distributed agile. They were able to get massive benefits because as a team they were working as one team where business and IT were working together as one team, collaboratively working. And some of the outcomes while they looked mind-boggling, they were possible because, as I said earlier, they were working as one team. One of the first benefits was speed to market. The average cycle time

reduced by about 300%, which means the cycle time came down from 7 months to 2 months. The new features. With the new features, the business was able to capture a bigger market and, thereby, increase their revenue for that revenue service market by 80%. The velocity of the team increased by over 50% on average. There were times where the sprint velocity was much higher than that. What was interesting apart from some of these benefits, which came out of the part of sprint retrospection was the Agile option showed that team camaraderie really improved. After every sprint, they would take a team morale survey. And what showed up was really that the morale, the collaboration was really improving in terms of how they could probably do things. So, as I said, these outcomes while they looked to be big and great, but they were possible because this particular team, the business and IT, they really worked as one team collaboratively, providing constant feedback and continuously looking at ways of improving. --Jeffson, thanks for those examples.

DevOps Benefits & Etsy

We've talked a lot about Agile, what we're doing to the help our clients get the value out of that. Let's shift a little bit and talk about DevOps. And Mirco, I think you are probably are leading expert to talk about DevOps, and I think you need to clear up some of these rumors that you've actually wrote the book on it. So maybe you can explain a little bit about DevOps and how we're really helping our clients in this space. -Absolutely. It would be my pleasure. A lot of the benefit that Jeffson described earlier actually rely on Agile and DevOps working together. The way that I always describe this to my clients is Agile helps you build a better product, while DevOps helps you build that product in a better way, cheaper, faster, more secure. And so together, you really get those synergies that allow you to have these amazing benefits that Jeffson described that are all about enabling business to get new product into the market quicker and being able to experiment and ultimately satisfying the customer. The really amazing thing when you look at DevOps in comparison to Agile is that DevOps doesn't have a manifesto. So there's no piece of paper that you could rely on to say this is what DevOps is, and that has led, in the last few years, for this content to expand. We know called it DevXOps because there's security included, which is DevSecOps. There's the business included as DevBizOps. There is the network included as DevNetOps, etc. etc. And what that allows us to do is really to focus in on a new dimension that perhaps was not as focused on previously. And that means when we find a new problem and in a transformational sense, there's always something new coming up that is the next barrier, the next blocker. So we can continue to expand the concept, and I think that's really, really powerful. What you can achieve with this is super high level of automation, of having these cohesive teams that Jeffson already spoke to in an Agile context. In a DevOps context, we the consider that a cohesive team across development and operations as in DevOps does. You can then really achieve things like having one engineer looking after hundreds, if not thousands, of servers. You can have the vast majority of problems in your systems identified through intelligent monitoring and the AI and machine learning so that there are only very, very few tickets at the end having to be treated. And then you can, the developer, the most expensive person on your IT team, he can be 2, 4, 6, 8 times more productive because he doesn't have to worry about any of the mundane tasks of the, you know, do I have to

check anything, do I fill in Excel sheet, or raise tickets because everything is automated. And that really allows IT to move away from this kind of old school production idea where, you know, you have this assembly line, and you just you know hammer away to really doing creative work, solving business problems together with the business stakeholder and an Agile team. So to me, that is super, super powerful. And because this transformation is so comprehensive and it has so many different aspects, it is absolutely correct that a few years ago I put my thoughts together in a book that really shares what my experience has been in this space. So yes, it is true. It's not the book, but a book about DevOps. -You know, Sarah, if I could, can I just talk for just a little bit longer about Agile and DevOps and why it's important we talk about these things together. You know, it's more than about being peanut butter and jelly. I actually like to say, especially for software development, that you can do DevOps without Agile, but you can't do Agile without DevOps. And to bring that to life, let me put this in the context of one of the true pioneers in this space, and a lot of people probably listening today know exactly who I'm talking about. I'm talking about Etsy. So Etsy is an E-commerce site that's best known for selling homemade goods. And they've been really instrumental in helping to bring Agile and DevOps practices to the masses. And one of the really neat things that Etsy does that I love is a principle they have in the organization called first-day deploy. Basically, this means that on your first day as an employee, you can write code and put it into production, and I'd like to know, I'd like to know how many people out there work for clients where that's possible or would be accepted. You know, they can do this because they have such fierce faith in their DevOps capability. And here I'm talking about the tooling underneath it, the tool chain, which guides the software development supply chain. It's comprehensive and fully automated. And the techniques that they use to embrace the very best of continuous integration and continuous delivery, in this case, I'm actually talking about a concept called canary testing where a new feature is released to a small number of typically friendly users, and any problems with it are actually benign to the larger day-to-day business of Etsy. This allows them to release new features in a controlled way so that if there is an issue with it, it doesn't cause a problem with the larger systems. So go back to what I was saying earlier about being agile means sacrificing quality. It means moving much faster, but doing it in a very protected way. So in summary, for Etsy in particular, Agile and DevOps are way, way beyond methodology. It's how they serve their customers better, how they forge stronger ties with their business partners, and it's about how they rapidly evolve their business to stay at a competition. So make no mistake. It is a misconception that moving faster sacrifices quality. In fact, it improves it through automated testing and isolation techniques like canary testing such that quality is better than you could ever obtain with waterfall. -Yeah, that's interesting. And I'm going to have a whole new perspective the next time I go shopping on Etsy. I had no idea. That is really interesting. And Mirco, no worries. Usually we have Paul on the show, and he's always shamelessly plugging his book. So we're used to it. We don't mind promoting your book. It sounds like a great buy.

Examples for Non-Techies

Okay, well this has been really interesting, and I've already learned a lot. So we've learned about Agile and DevOps. They're way more than a methodology. I mean, it really is a different way of working, a different way of thinking. And we've seen some great examples of bringing value to our clients and how we actually use it ourselves here at Accenture. So I'd like to transition to one of those other misconceptions you talked about, Adam, where you said that it's not only for software development and that we can use it in other aspects of our work. I'm really interested in this one. -Sure. You know, while the example that I gave about Etsy is all about software development, and that's where these principles were born, the truth is the underpinnings of Agile and DevOps are actually being used well beyond software development these days. You'll even hear Julie Sweet talk about this, for example, when we're rolling out the next generation growth model. But don't just take my word for it. I'm going to have you listen to for a moment Allison Horn in HR. Yes, that's right. HR, human resources, is a big fan of Agile and DevOps and was an authentic and enthusiastic early adopter in Accenture for applying Agile and DevOps to a non-software development environment. Allison, do you want to tell us a little bit about that? -Thanks, Adam. That's right. We have adopted a lot of the both mindset and practices across Agile and DevOps within our global talent organization and even more broadly across HR. As we talked about here today, Agile and DevOps is so much more than just structures and tools. It's that mindset that way of working. This is what we brought to life across all of our teams. These global teams lead with collaboration with our customers, collaboration across the different functional areas from the start, iterative design and development throughout, for MVPs, continuous improvement front and center at all times. And while all of our teams have absolutely adopted this mindset, most of our teams have also adopted at least a portion of the Agile structures and tools, things like daily stand-ups, Kanban boards, retrospectives, and so forth. And we even have about half of our teams working in full scrum mode with rigorous sprint management across. So the TQ team, the team that brings you these programs and the rest of TQ, that's actually a great example of a team that's adopted the mindset 100%, but has chosen to adopt only portions of the model, portions of the structures here. So this team keeps the TQ program on track through a daily stand-up meeting. Every aspect of the TQ program, from content development to tech integrations to communications, all of it lives within an extensive Kanban board. We actually leverage Planner from Microsoft Teams right here. So going back to how we got started here, this was about 2 years ago. We knew it was going to take us more than just a few tries to get this right. At the time, there weren't a lot of examples for us to follow with respect to how Agile and DevOps, or perhaps what we would call DevBizOps, can be applied outside of software development. We benefited from some amazing advice from our friends in CIO. They were already well into their Agile journey on their own, and their recommendation was that we try this out with just one or two teams first before we bring it to the whole organization. And this was such great advice because it did take us a while to get to a point where we felt like we'd gotten this right. We had to start with the questions about, you know, for which teams, for which kind of work would this makes sense all the way to which of the tools and the rituals would work best for us, particularly since we continue to work in a fully diverse virtual environment. And the pioneer teams that went forth for us really help to set the stage for a much broader expansion of Agile across the whole organization. So let me try to really bring this to life with an example of one of our pioneer teams, which is our sales learning teams. They shifted to scrum about 2 years ago, and they

were hugely important with respect to us figuring out how we were going to make this work in a non-software development environment. So prior to that shift, they followed a very waterfall-esque process for course development where one team would gather requirements, one team would do the design and development, a third would come in for delivery. And a typical master class would take about 16 weeks to cross the finish line. And it was rare for a first run of that course to go as well as we would want it to. And it was rare because the people who are responsible for course delivery were missing a lot of the context that had come in through the design and development phases. And the people who designed the course likely weren't doing so with all of the delivery challenges in mind. So kind of a typical issue with, you know, siloed teams passing things over the wall and not necessarily having all the contextual understanding that would have come otherwise if they'd been working in more integrated ways. So fast-forward to today. That same class goes from start line to finish line in 8 weeks, half the time it used to take. And the number of issues we see in the first run, they're down by 75%. We have the exact same number of people working in the space, but they're working very differently. The scrum teams have people from each area working throughout each phase. Everyone's engaged from requirements all the way through delivery. They work in 2-weeks prints. They ship a product at the end of each sprint, and every retrospective, even 2 years later, continues to yield new insights that get applied in the next sprint. So I mentioned the benefits with respect to speed and quality. And while those are definitely important, this way of working has brought even more important benefits forward with respect to prioritization. So I can go forth and we can increase the speed, and we can increase the quality of a product. But if that's not actually the most important product to be focused on in the first place, then we really haven't delivered as much value as we could have otherwise. And the reality is this agile mind set and framework, with all of its structures and rituals and tools, it provides a path for transparency of demand that we just simply didn't have before. As an internal HR team, we don't create statements of work for our services. There is no such thing as a change request process when Adam calls us and says hey, I need to add yet another tech learning priority to the urgent list. Instead of the 27 that you're working on today, I need to add another 10 to those. It just doesn't work like that. Maybe we should have a change request process for Adam, but we don't. So in yesterday's world, we get the call from Adam, and we would say okay, we understand. This is another new set of urgent priorities. We will add it to the additional workload. And every time we did that, we just slowed everything down because, you know, there's a lot of things we can fix. But one of the things we can't is the limitation of the 24-hour day. So we would get into this place in space where we would be trying to move so much forward at once that it was like a snail's pace at every single piece. But with Agile, each team keeps a very transparent, detailed backlog of the work that's been requested, a plan and a commitment for the work that's going to be delivered in the 2-week sprint. And as these new situations arise and come in, the teams are able to have those either/or conversations that were really missing from the way that we worked before. They sound like this. Yes, Adam. We can take on that new work If you want to pause another project that we currently have underway. Or, you know, we can take that on as long as everyone understands that this is going to push our sprint out a few extra days. Or better yet, how does this compare to what we're working on now because perhaps we just prioritize what you're asking for today in the next sprint and so forth. And I'm making it sound easy, I know it's more challenging in real life, but the point is that

working in this way has made it far easier and more transparent to talk about priorities and make sure that we have the team focused on the most important products at the given time. And those constructive conversations about tradeoffs help us get to this point of focus and transparency that are really crucial in the world that we're living in today where requirements are changing constantly. -Well, Allison being in CIO, I can certainly attest that we're doing Agile and DevOps for all of our application and technical environment work we have. But I love this idea of applying these concepts outside of a traditional technical environment, and I'm really excited to think about where else we could apply this. And honestly, it's not surprising that Accenture is applying both Agile and DevOps outside of just our technical work. It really does seem that there's not really a limit how you can apply these concepts if you just use a little bit of creativity.

Q&A from 'Learners'

So, I think in the spirit of being customer obsessed, it's time we check in on our live Twitter feed and see what our learners are saying and asking about Agile and DevOps today. -Sure, Sarah, hang on for a second. I'm having problems with a couple things. First of all, I didn't submit 27 different new learning requests, it was actually 38, and Allison, you should probably check your email. -I'm sure we've got it now, we're good. -That's Agile. And listen, you know, Sarah, you told me that this show was recorded so that any bloopers or other things we could do could be pulled out. So, I'm confused how we have a live Twitter feed for a show that's being recorded, so maybe you can enlighten me. -Yeah. We just told you it was recorded so you wouldn't be so nervous. This is all live, and we have a very live and active Twitter feed. And since you are so interested, I think you could take the first question. So, let's go with this and see what the first question is. It's from A. Jyle, and they ask, oh, great question, does Waterfall still exist? And when would an organization or company choose to do Waterfall over Agile, and can you mix the two? -Hmm, somebody's name is actually A. Jyle. That's interesting. And they asked a great, great question. So first of all, Waterfall absolutely still exists. In fact, it's the foundation for how a lot of our clients still deliver software projects today. However, Agile is becoming more and more prevalent and more and more common. Now, it doesn't mean that you can do one and not the other. There are lots of clients around the world because of the way their software is built or other ways that they're using Waterfall to maintain it, and in other parts of the organization they're doing Agile at the same time. Sometimes even on a project you'd use both. In order to get the MVP release, sometimes using a Waterfall technique is the best way to actually get yourself to the point of having a minimum lovable or minimum viable product, and then you can switch to a more Agile delivery method afterwards. So it's about doing what's right for the client culturally, what their software is capable of doing, and otherwise. And I find that these things are oftentimes mixed together, although the future is definitely more Agile and DevOps, and we see that becoming more and more the dominant approach for how we maintain and deliver systems going forth. -Thanks, Adam. You did a pretty good job with that question. Now, Mirco, this one's going to be for you. Our next Accenture Twitter user, X.O. Sistem, asks, what is our ecosystem partner network like for Agile and DevOps? -What a fantastic question. I mean, in this space, there's a new tool and a new organization coming up every month. So, how do we deal with that? The reality is that with a lot of the

Agile Frameworks, we have really good partnerships. Because, if you think about it, we work in some of the most complex, largest, most technology-comprehensive organizations in the world. So, if you want to really stress a Scaled Azure methodology, we have a fantastic place to do that. And that's why I think our partners really appreciate that we're working together with them, and we provide a lot of feedback into them. For example, for the Scaled Azure Framework, we are one of their gold premium partners with some of the largest amount of coaches and certified people in any organization. And the same is true for a couple of the smaller ones, and in the DevOps space, we've partnered with the DevOps Institute, for example, to provide industry acknowledged training. Because at the end of the day, some of the training that we provide our people should be the same that our clients get so that we all speak the same language and that we have certifications that are appreciated on both sides. On the tooling side, we work with a lot of different vendors, but we are having a focused area on the stuff, on the tools that are really helping in complex environments. So, for example, with Atlassian, we work on how do we deal with Agile in a really complex environment? How do you manage all these different team setups, but then still aggregate information that you needed to put for your level so that you can manage the flows of budgets and the managed complex scope? In the DevOps space, there's a lot of tooling coming up, and there's a lot of consolidation and new tools being invented again and again, and our focus there is really on getting our people skilled up to be able to use those tools. So we're working with organizations to run trainings for us. We have them come in and present new tools and really help them evaluate whether we see value in that or not. And then the other thing that we do with our partners is really going to market and help shape the perspectives on DevOps. So, for example, we did with Red Hat recently write a white paper about operating models in DevOps and how you really enable innovations for that. And we're speaking at webinars and conferences together with our partners. And we even do joint solution architecture sessions where we just try to come to a room, take a complex problem, and see how we can solve it with their tools and with our methods. And that creates a very, very rich tapestry and fabric of really partners. It's not really vendors, it's partners on how we can solve these transformation of problems together. -All right, well, the questions just keep coming. So, John, I'm going to throw this next one to you. This is from our Twitter user named Saul Eye Queue, and they would like to know, what does Accenture do with our clients in the spaces of Agile and DevOps? -Saul's actually a very active blogger. I follow him as well. So actually, we're kind of in a very interesting time at Accenture. Neville and myself came over via acquisition of SolutionsIQ just a little over three years ago, and that was part of Accenture's desire to kind of expand its capability around helping in an advisory space in the Agile space. So when you take Allison's story, and by the way, Allison, I am now taking you to every new client visit that I have because I couldn't get a better testimonial. -Well, you could, because I should actually tell you, SolutionsIQ was our partner in bringing us forward, so I didn't even mention that before, but we could never have done what we did without SolutionsIQ. -I was waiting for that plug, so thank you. -There you go. -But it's exactly that, and I think what Allison did, she really described kind of the journey very well, and when we come in in an advisory space, what we're doing is it's less about here's how you do it, follow these guidelines and everything is going to be good, and it's more about what are you trying to accomplish? Where's the vision for where you're going? And we have a tool kit of kind of best practices or patterns, and then from that, it's an assembly of what applies for the organization

to make sense. So, this is a capability that Accenture has, and we fit well with our friends in strategy and consulting as they help organizations on their journey. We can come in, and in particular, you hear the term transformation quite a bit, you hear about cloud transformation, you hear about data transformation, you hear about digital transformation, you hear about Agile transformation. Well, this is all now starting to swirl into one overall value profit our clients are looking for. And that is, how do we build a responsive organization? How can we be adaptive in what we're doing? And how can we actually kind of create a continuous innovation engine? And in all of those scenarios, an Agile backbone as an operating model is absolutely essential. So, we can kind of bring to bear this advisory practice. So we're, you know, you say, it's a really great time at Accenture because we've got all of these different capabilities that have now ripened at the point where we can, I think, probably better than anyone in the world, come and provide solutions that are end to end, that are going to help organizations get that responsiveness. -Yeah, that's great, and really bringing the best of what we have from across our enterprise to our clients. That's great. Thanks, John. All right, Neville, I think this next question would be perfect from you. This is from a Twitter user named Finn Sector, and Finn wants to know which industries or markets seem to be the fastest or the slowest to adapt these practices? -That's a good question, Finn Sector. So one thing that is pretty common is that most organizations at this point have dabbled in Agility. They either have a couple of teams or they have a program that is Agile, and maybe their e-commerce group was just traditionally where some organizations tend to start. But I would say most, you know, 95% of organizations across the globe are having some level of Agile goodness inside the organization. Now, you have the Googles and the Facebook and technology companies of the world that just grew up Agile. You know, there was nothing to really adopt because Agile was just the way that they worked, but there are some organizations that seem to have adopted Agility faster. I would say some of the financial services organizations and insurance companies were some of the first organizations that I personally worked with 12 years ago starting their Agile journey. But companies that tend to be the laggards are, it seems to be more of the resources clients, some of our energy and gas and electric companies are a little bit slower to adopt these practices, primarily because of the massive amounts of regulation that they have to adhere to, which makes the adoption of some of the principles a little bit more hard, but we do have organizations that have adopted these principles inside of the gas and light industries. So that's, you know, the financial services companies kind of kicked it off in some regard, but what you're also seeing now over the last few years is retailers are really starting to respond, or they started to respond to the burning platform that Amazon created. So retail companies had to pivot fast. How do we get feedback from our customers and get that new product or new service out the door quicker than they were in the past, simply because of the urgency that organizations like Amazon created. -Yeah, that makes sense if you think about some of the cultures and the ways of working in some of those different industries. Thanks, Neville, that was great. All right, Allison, I love this question. I'm going to give this one to you. And this is from a Twitter user named Miss Conn Sepshun, and Miss Conn Sepshun asks, if I make a mistake, I can just blame it on Agile, right? What do you think about that, Allison? -Absolutely. You can blame it on Agile. You can also blame it on the weather, you can blame it on the stock market, you can blame it on all kinds of things. But it's not an appropriate place to put the blame. Because the reality is with the mindset and the structures and the tools that Agile brings forward and the

transparency of all of the different teams working together, you actually have more visibility to what's happening across the team at any given time than you would have otherwise, and that's super important because mistakes happen. We all make mistakes. Mistakes arise through every single product or service that we bring to life. And working in this way gives you the opportunity to see those mistakes as they're starting, as they're emerging. You get to actually have increased visibility to things that are going off the track and you get to reroute them, reset them far sooner than you would in a more traditional Waterfall way of working. -Okay, this, I think, is our last Twitter questions. So, Jeffson, I'm going to send this one over to you. And this is from our Twitter user named Sam Kanbanopodopolous. Yes, that is his real name. And Sam asks, what is the difference between Agile, Scrum, Kanban, et cetera? I thought they were all Agile. -These are all frameworks or ways of working, and they all fall under the umbrella of Agile. They all implement the values and principles which Agile talks about. Now, so the next question typically should be asking is, okay, if I'm on a project, which Agile methodology should I be using, or which Agile Framework should I be using? You should be looking at, for that particular question, you should be looking at what is your business requirement? And what is the context in which your team operates? Now, typically, for example, if the real business requirement for engagement is to really reuse their time to market, you'd probably pick up Kanban. If the requirement, if you have a large project, and then that has to be, you know, stripped incrementally, then you would look at Scum. Today, a lot of our projects are combining Scum and Kanban and call it Scrumban. So it typically, as I said earlier, depends on what type of engagement you have, what are your business requirements, how are teams working together to really decide what goes best for you. -Wow, thanks everyone. I think you all did a pretty good job of fielding some pretty tough questions from our very real Accenture Twitter accounts.

Book Recommendations

As we always do, we'd like to end the show with some advice from our experts. We talked a lot today, a lot about different topics related to Agile and DevOps. And it's always interesting to see what our top leaders, like all of you, are actually reading as it relates to the topics we're discussing. So let's close with each of you telling me your favorite book on Agile or DevOps and why. So, Neville, let's start with you. Since you coach our clients daily, what is your favorite book? -So the book I'm reading right now that I've actually shared with a couple of my clients is called Leadership Is Language by David Marquet. And the reason why I like this book, as it relates to Agile, is because in order to be successful in any transformation, leadership has to demonstrate behaviors that they are expecting of their teams. And Leadership Is Language is one of those books where it really helps kind of bring that to light and helps leaders realize how they can enable or become an impediment in any type of transformation. So that would be my book, Leadership Is Language. -Great, that sounds great. All right, Jeffson, what about you? To be honest, picking one favorite book is actually a big, big ask. But, really, there are so many great books on Agile DevOps. But I will pick up two books which really have made a profound impact in the way I think about Agile. The first one was and is Coaching Agile Teams by Lyssa Adkins. It really brings up the essence of Agile in terms of coaching, in terms of how teams should behave, in terms of that collaboration, empathy, all

those aspects. Great book! My second book and which I consider as probably my Bible as well is the Lean Software Development, which is by Mary and Tom Poppendieck. It really brings out the practical aspects about how really we should be thinking about our Agile mindset in terms of ways of working, in terms of continuous improvement. Brilliant books. These books have really been books where I really have learned a lot. -Okay, great. Thanks, Jeffson. We'll let you get away with more than one recommendation. John, how about you? -Well, I'm going to take the liberty then as well with multiple recommendations. So first of all, I would say this actually predates the actual movement. It's called The Goal, and it's about constraint theory, and it's kind of this---its constraint theory in a cheesy novel. So it actually makes it really easy to understand these base principles. But I think when you get into the Agile space, this is like these are good fundamentals for somebody to kind of understand. It's a very easy read. Steve Denning pretty much anything he's written in the last 10 years, the most recent, I think his most recent is The Age of Agile, which kind of summarizes where we are and how this has evolved to where we are today. And then the last, just because I want to beat Jeffson so I'm going to come up with three here, the author of The Black Swan, and I can't recall his name right now, he wrote a book called Antifragility. And what this is is this is the concept of leaning into uncertainty, right, not being overwhelmed, not being---but actually when you recognize that this uncertainty is now current state, how do you take advantage of that? And I think that's actually a great mindset for people to kind of start to explore and start to drive in that direction. -Yeah, great, and can be applied to so many things that we do. That sounds like a great read. Okay, Allison, you don't have to have five, but what would be your recommendation? So this is the problem of going last towards the list because this is the book I wanted to talk about, which was just mentioned, a Steve Denning book, The Age of Agile. If you can see the side of this, I have gone over page by page of this book. And for me, this was really the book that moved me to action to say, Okay, we can't NOT do this. So this is a book that I recommend to anybody looking to get into this space. I will do a small plug to say if you're not into reading a book this thick, there is an abstract available through an offering that we have for everyone here at Accenture called Get Abstract. If you go to My Learning and just search on get abstract, you can get an abstract of this book and get 15 to 18 or so pages of it and get a lot of the key concepts in there as well. So you can even be a little bit faster and more Agile with your reading. -Oh, that's taking me back to my college days. Thanks, Allison. And Mirco, what about you? What book would you recommend? -By pure coincidence, I have this book lying next to me that I would recommend to people. So this is an award-winning book. As you know, it was the DevOps book of 2018, so absolutely brilliant book. The author really knows what he's talking about. There's a lot of really practical advice in here, and actually I would go one step further. I would say everyone in Accenture should go to Amazon right now and purchase a copy of that book. It's called DevOps for the Modern Enterprise. -Yeah, okay. Thanks, Mirco. I mean, I think it does sound like a pretty good book, so I think several of us will probably go out and buy that, even though it's your own book. Okay, and last but not least, Adam, what would be your recommendation? I have to say we have an incredibly well-read panel here, but I'd be remiss if I didn't add my own book recommendation, and it's really a great one. It's called The Phoenix Project, and it's by a person named Gene Kim. What's great about this book is it's written as a narrative, so it's a story, and anyone can enjoy

it. And, actually, it's sold more than half a million copies, and I've personally purchased at least 50 of these because I've given them to my clients. And, by the way, you're welcome, Amazon, for that.

Agile & DevOps Closing

-Let me close now by just thanking all of our panelists today. We've had our genuine experts at Accenture talking about Agile and DevOps for you. And I'm always proud to be in the same virtual room with them. And for you the student, yes, I'm talking to you, thank you for investing your time to absorb this content. Agile and DevOps are without a doubt absolutely mainstream now, and they're not just for IT. -Well, today has been an amazing show, and I think we did top a record for the number of guests that we had. So thank you to all of you, Jeffson, John, Neville, Mirco, Allison, and most of all, Adam. This has been a really great show where we've learned a lot about Agile and DevOps. But there's always more to learn, so if you want to learn more, go to the Go Deeper section on Pluralsight and our TQ homepage, where you can find links to great Agile and DevOps materials and case studies. And we'll always see you next time. This is Sarah Dugan signing off from our virtual TQ HQ.

Agile & DevOps in Review

What is Agile & DevOps?

What is agile? Imagine you have a product to develop on a short timeline. In a traditional gated process, you make predictions about the requirements you have and then construct a plan around them. Using this plan, you sequentially move from one phase to another. You design, you get approval, you implement, you get approval, and when it's all done, you finally test it and deliver it to the customer. It turns out your predictions were slightly off, and your product needs to be redesigned, but you can't because you're on a short timeline, and you already spent your budget. You'll have to deploy the imperfect product as scheduled. In an agile workflow, you work in short sprints and adapt your plans as you go based on your current situation. This opens up your entire development world. Your team can take what they learned from one sprint and apply it in the next sprint. The product can get refined and improved iteratively as new requirements and new constraints emerge. Agile lets teams design and develop together, testing all the way with customers. When the product is done, you know it works because you worked with an agile mindset. What is DevOps? When it's time for products to be deployed, teams from all over the business and IT need to work together. Often delivery methods need to be adjusted for proper output. That's where DevOps comes in. They're responsible for automating delivery methods, infrequent releases for maximum efficiency. When DevOps teams work with an agile mindset, the business results dramatically increase. This means more quality products and services to customers faster, more efficiently, and at a lower cost. This is agile and DevOps.

What Does Agile & DevOps Do?

What do agile and DevOps do? They help businesses deliver high-quality work to the market faster while always keeping the customer in mind. Agile does this by encouraging us to adopt new ways for working such as delivering value early and often, prioritizing and focusing our efforts, delivering the highest value first, working in small batch sizes, working incrementally and iteratively, pulling quality forward, working transparently, inspecting the work and adapting as we go, and encouraging agile leaders. DevOps does this by focusing on continuous integration and continuous delivery, or CICD. Working in tandem, agile and DevOps make it possible for businesses to delight customers with great products that solve their problems. What do agile and DevOps do? They help us deliver the right work to customers faster and with better quality.

Why Does Agile & DevOps Matter?

Why do agile and DevOps matter? Early adopters of agile and DevOps have become well known for disrupting and forever changing their industries. Uber upended the transportation industry without owning a single vehicle. Airbnb just cut a hole in the hotel industry without owning rooms or hotels. Netflix ushered out the big players in the movie rental industry without any brick and mortar stores. Traditional market leaders are feeling the pressure to become more agile. Leading businesses must innovate and deliver more effectively than their competitors while being willing to disrupt their own products and business models to survive and thrive into the future.

How Is Agile & DevOps Applied?

How are agile and DevOps applied? Here are four ways agile can be built into the way a company works. Number 1: Breaking projects into smaller pieces. 2: Using visual management. 3: Limiting the work in progress. 4: Working with engineers. Here are six ways that DevOps can improve the way a company works. Number 1: Decreasing time to get the end product running in production. 2: Increasing feedback from production back to development. 3: Continuous learning to improve and evolve processes. 4: Boosting team engagement. 5: Creating more effective leaders. And 6: Increasing customer satisfaction. Successful use of agile and DevOps results in more products and services out to customers faster. This creates a steady flow of reduced costs and increased profits. Agile and DevOps are on the move, changing entire enterprises for the better.

How Does Agile & DevOps Work?

How does agile work? The short answer: Small, cross-functional teams work together to make things happen. The long answer is more complex. Small, cross-functional teams use one or many agile frameworks to help them prepare and do their work. There are many different agile frameworks out there, but two of the best known our Scrum and Kanban. Let's start with Scrum. Scrum is a way of working in which the team works in a series of short bursts known as sprints. Each sprint ends with the team delivering some small piece of finished work. Sprints are carefully planned so everyone on the team has a specific task to do and enough time to complete it. At the end

of the sprint, the team shows off its work and gets feedback that will affect its plan for the next sprint. Kanban, on the other hand, is a way of visualizing the work the team does. A Kanban board illustrates all the things the team is planning to do, is doing, and has done. It's a snapshot of the team's progress at any moment. Kanban forces the team to think about how much work can be reasonably done within the sprint timeframe and then limit their work in progress. This helps prevent bottlenecks and boosts the team's focus. Scrum and Kanban are often used together, but there are other agile frameworks to choose from too. One framework is XP for extreme programming. It focuses specifically on software development. Another framework is SAFe, or scaled agile framework. It's a set of workflow patterns that combine Scrum, Kanban, XP, and DevOps into a super framework that can be scaled up for a large organization. No matter which framework your team uses, a team that embraces agile values and principles will create work that delivers value early and continually improves.

What is Accenture's Role with Agile & DevOps?

What is Accenture's role in agile and DevOps? Well, every single part of our business incorporates agile and DevOps in some way. Let's look at the media campaign we created for SolutionsIQ. Shortly after launching the campaign, we learned the logo could be improved to enhance the brand presence. Working in an agile way meant that we could sense the need to make this change and then apply it early on instead of after 40 different social images had been created. And on the DevOps side, Accenture currently partners with over 150 different companies to speed up our delivery of work. We work with over 40 market-leading software vendors for top of the line development. This means customers have access to powerful apps and tools for little to no cost. And on top of all this, we provide training and insights into those tools and apps that will drive your business to the next level.

How Does Agile & DevOps Combine With Other Technologies?

How do agile and DevOps combine with other technologies? Agile and DevOps can be used in conjunction with almost any technology. For instance, using cloud with agile and DevOps allows teams to finalize features and move them to the cloud for instant feedback. Artificial intelligence can transform DevOps with enhanced software testing. It can liberate and organize data. It can predict errors before they cause problems. And it can improve the collaboration between development and operations teams. Teams can also incorporate security with DevSecOps. And for added security, you can even use AI with DevSecOps to detect hackers and automatically record potential threats. Agile and DevOps help you work with different technologies in exciting and powerful ways.

TQ Agile & DevOps Wrap Up

TQ Agile & DevOps Wrap Up

Well that's what we have for you on our core Agile and DevOps content. I hope you learned something new about Agile and DevOps and you're feeling more confident that you can talk about it with your teams and clients and, even better yet, start to put these principles and mindsets into work for your area of the business. If you're looking for even more learning on Agile and DevOps, visit the Go Deeper channel here in Pluralsight or return back to the TQ home page and see the other resources and case stories that we have there. Make sure to tune in again soon for our next topic. Until then, keep growing your impact and growing your TQ and happy learning.

Course author

Accenture

Accenture is a leading global professional services company, providing a broad range of services and solutions in strategy, consulting, digital, technology and operations. Combining unmatched...

Course info

Level	Beginner
Rating	★★★★½ (871)
My rating	★★★★★
Duration	1h 46m
Released	21 Jun 2020