EPISODE 900

[INTRODUCTION]

[0:00:00.3] JM: Airlines have always had an emphasis on new technology. Over the years, airlines have needed to develop more and more software. Digital transformation is causing every large company to adopt the tools and practices of software companies and that includes Delta Airlines.

Delta Airlines has existed for more than 90 years. Over that period of time, the company has developed new systems in every generation of software, from the days of mainframe computers, to a modern Java-based backend. When the DevOps movement started to take shape, Delta Airlines started to take a more focused approach on continuous integration, version control and an organizational structure that removed silos between teams.

Jasmine James is a Manager at Delta Airlines, where she focuses on improving the software practices of the company. She joins the show to talk through the process of changing the developer culture within Delta, as well as what it's like to build software for an airline.

Jasmine is speaking at GitLab Commit in Brooklyn on September 17th, 2019. GitLab is a sponsor of Software Engineering Daily. If you're thinking about attending GitLab Commit, where Jasmine will be speaking, you can go to softwareengineeringdaily.com/commit and enter code COMMITSED. That's COMMITSED and save 30% on your ticket, while also supporting Software Engineering Daily.

I hope you enjoy today's episode.

[INTERVIEW]

[0:01:39.2] JM: Jasmine James, welcome to Software Engineering Daily.

[0:01:41.4] JJ: Thank you so much for having me, Jeff.

[0:01:43.4] JM: Absolutely. You work at Delta Airlines. What kind of software does an airline need to build?

[0:01:50.0] JJ: Well, you think about a traveler's interaction between the whole journey, or throughout the whole journey, and all of that is supported by technology from checking in, bag being tracked, the process of them boarding a plane. Pretty much anything that you could think of, any type of technology, RFID, we are leveraging it here. Lots of software.

[0:02:14.6] JM: Delta has been around for more than 90 years.

[0:02:18.2] JJ: Yes.

[0:02:19.5] JM: Are there people at the company that are still working on the legacy software systems?

[0:02:23.5] JJ: Absolutely. A lot of the backbone of the technology that I just mentioned is supported by legacy systems, like TPF, lots of different things like that. We still have those mainframe folks that are out there developing and actually adopting a lot of the more modern processes and mindset, even within that legacy infrastructure, so it's really, really cool.

[0:02:49.1] JM: Really. How much time do you spent interacting with the mainframe folks?

[0:02:52.9] JJ: Well, we are very early in our DevOps journey right now. We are just getting them started with the process of a version control, those types of things. It's a slow process to adopt all these things, especially if they've been working in one way for a very long time. I would say, probably about 10% of our time is spent with a lot of that legacy development teams.

[0:03:14.7] JM: If you're developing that legacy mainframe software, can you write that software on a MacBook, or on a on a Windows machine, or do you need to be on a specific older type of computer?

[0:03:26.8] JJ: Yeah. It's all limited, right, to the technologies that are still legacy, right? We're not able to leverage a lot of the more modern operating systems, like Mac OS, or Windows 10.

They are actually pretty much utilizing the newer methodologies exactly where the legacy – legacy OSs are so in place.

[0:03:47.2] JM: That sounds pretty tricky. That's a constraint I have not heard explored. Can you tell me about how – I mean, to what extent you've been able to port those modern processes to the older systems?

[0:03:56.9] JJ: Yeah. We're really at a basic level right now. We're wanting to do basic things, like let's get version control, right? That's pretty much a basic capability of DevOps. Let's try to figure out how we can version code, so that way you're not really stuck at a certain point within your development environment, right?

You can actually have those thing versioned, they could be using – it's older version control, but let's just get version control and let's get automated builds happening. those are the two things that we've decided to attack first. Once you get that build, even if we'd have to deploy it manually, at least we have version control in the build process automated. It's some value that we're offering them.

[0:04:38.1] JM: What's the interaction point, or the integration point when you have the older systems, the older mainframe developers, their software must eventually interface, or touch base with the newer Java-based systems. What does that integration surface area look like and how do those different teams interact with each other?

[0:05:01.5] JJ: Right. As you probably can guess, we have a lot of different services that are leveraged here at Delta. Without getting too specific, usually it's done through Scylla calls, or things of that nature that interface with legacy backend system. We're just working on modernizing those calls and creating APIs that serves those same capabilities. That way, we could have the new more modern development methodology is the bridge between the legacy ones, so through Scylla and APIs.

[0:05:32.6] JM: This is an airline. This is not a company that's creating software that just displays cat videos, or something trivial, are there different constraints around these kinds of systems? Do you have to think about safety concerns and extremely high-reliability in this

software? Or what's the level of safety and security that you need to perennially be worried about with building a airport-related software?

[0:05:58.9] JJ: Right. We have, of course different categories of types of applications that are being leveraged all across the airline. Our most critical systems are things like okay, how can we make sure that everyone is on a plane, or the safety checks have been done? Those are absolutely and necessary for us to operate, right? We are a regulated industry. We have to answer to the FAA and other entities for all these things. Those are the ones where we make sure that we have scalability and they're very reliable and resilient system, so that way, we don't have to enact a ground stop if these applications become unavailable.

Other types of applications, there's a little bit of wiggle room for the availability and resiliency, but we do have some applications that are categorized as mission-critical that we absolutely must have. Then we also have concerns, like everyone else when it comes to customer data and making sure that restoring that properly. We of course, have safety policies around that as well.

[0:06:58.5] JM: There's a tension in software development that unsure how to resolve. That tension is that in some context, it seems like companies that are able to iterate faster are able to develop software that is more secure and safer. I can imagine that some domains, or some particular problems within domains, such as flying an airplane, you would want the software development process to purposefully move more slowly, because there's a particular safety that you want to ensure through a slow development process. What's your opinion on the relationship between speed of software development and safety?

[0:07:39.8] JJ: Well, I think of where we've been as an airline and capabilities we've been able to provide through technology thus far. In the past 90 years, right, that Delta has been operating in this space, we've delivered software that enables us to run an airline pretty efficient airline. Right now, our focus is how do we speed up that delivery of software without pretty much giving up that quality, or the safety that you've just mentioned?

I think that as long as we've identified the processes and the checks and balances and we've considered all of those throughout the process and we've made sure that these are our gates

that we cannot essentially skip over, I think that as long as we keep that first and foremost, you can absolutely speed up. Never skip over those things that are super important, because the life, the airline depends on it.

You think about a lot of these safety – pretty much things that have happened in the news lately, right? That is something that could happen if you just for example, don't implement one of those processes that are currently in place. Yeah, absolutely you have to keep those first and foremost. I do think there are opportunities to speed up where you can.

[0:08:58.7] JM: Delta has been moving towards DevOps. That's a term that's hard to define. What is your definition of DevOps?

[0:09:07.2] JJ: DevOps. Wow, nobody's ever asked me that. My definition of DevOps would be to provide pretty much a collaborative development in operations in one; bringing it all together within one development team and breaking down those silos, essentially. A lot of people equate DevOps to mean tools. I think that tools are an important enabler of DevOps, but I ultimately think that it's a cultural thing that we have to break down silos and change the mindsets of individuals and bring those two things closer together in order to deliver more efficiently.

[0:09:42.5] JM: When did Delta start going towards DevOps?

[0:09:47.9] JJ: Well, we've always had DevOps in a sense. In some ways, there was collaboration happening within pockets of the enterprise. We made it a priority to pretty much enable teams to adopt DevOps more easily around two and a half years ago. The way that we did that is by bringing in some new tools that made it easy for us to implement continuous integration and continuous deployments here at Delta. That is about two, two and a half years ago is when we've really, really started to invest in the tooling that would enable us to have that cultural adoption and get that mindset going across the enterprise.

[0:10:27.3] JM: There are often a large number of behavioral changes and technology changes around DevOps. Trying to do all of these things at once can be overwhelming. How has the Delta organization sequenced all the different changes that need to be made within the movement towards DevOps?

[0:10:49.2] JJ: Right. When we started, we knew that the people part would be the hardest. We tried to one, build a foundation, which was bringing new tooling, getting folks educated on what this tooling was and the value that it brought to the company. Once we did that, we wanted to spark a grassroots movement, because you always have early adopters, right? Those folks who are really excited to utilize these new capabilities. We wanted them to take these changes back to their team and say, “Hey, we've been doing things in this way. Why not try to leverage these new tools and work a little bit differently and more efficiently?” That's our way of trying to make that move, granted there's no one-size-fits-all, or blueprint to how to do this.

It worked for us, granted we still have a long way to go. I think that what we saw was okay, those early adopters, they went into their individual organizations and they started to think about these things and have the conversations and really question, “Okay, what efficiencies can we improve by bringing these into our space?” I think it really sparked it as we intended.

[0:12:02.9] JM: Describe the software development organization in terms of its size and the breakdown of the different roles within Delta. I guess, the most important question is probably how big it is and how those employees are broken down into teams?

[0:12:20.4] JJ: Right. I don't have an exact number of developers, but I do know that within IT, I last checked, I think we were right around 4,000 people. The way that we're structured here is around business line. There's multiple areas of the business within Delta. We have IT that is structured in the same way. That way we can make sure that the business value for each of those lines is pretty much implemented right within their – in their space and you don't have to compete for that.

Within those individual business lines within our IT space, you have different development teams that work on applications within that portfolio, if you will. Pretty much, we're structured via business lines and then individual development teams within those larger portfolios.

[0:13:08.6] JM: How do the teams historically, I would say, how have the teams interacted with each other, the different teams? Because you talked about the silo-busting nature of DevOps. You obviously want to break down the silos, but historically you have teams in particular

verticals. Can you just describe the nature of the different teams and how they interact with each other?

[0:13:33.7] JJ: Yeah. Of course, there's different types of interactions across the portfolios. I would say from what I've seen, most teams only engage when it's time to handoff, or at some milestone. After something, some epic has been completed and it's time to pretty much let the organization know about it. Historically within any big organization that hasn't started to transform, you have those throw it over the wall moments. That was absolutely what we had here.

Here lately, we are and we're trying to get to business agility, right, and collaborate more across teams. Also within the team, be more cross-functional. That has dwindled down. The latter behavior, I just mentioned has dwindled down. Individuals are actually seeing the importance of collaboration early and often, because and historically, right? When you throw things over the wall to the next team, there comes – they have to scramble and try to figure things out. When you collaborate early and often, it makes for much more seamless experience across the board.

[0:14:42.6] JM: Why is that? Are there any examples you can give of why it would be productive for teams, or how it's been productive for teams to cross-functionally collaborate?

[0:14:51.9] JJ: Absolutely. One specific example that I can think of, large multi-portfolio efforts. I'm trying to say it without saying too much, but just bringing individual development teams across multiple areas together in order to deliver a feature. Historically, you would have one team develop something in their silo, the sister team develop something else and then the brother team develop something else. Then we just meshed it all together and hope it fits.

What we've done more recently is bringing those teams together from the inception of the idea, right? That way, you have an understanding of all of the technical risks, any implications that developing this capability would have on the organization, because you have everyone's perspective right then and there. This actually made the implementation that much quicker, because you knew everything upfront, right? You weren't just building things as you thought they should be built. You were taking into consideration to the other teams on perspectives and making changes based on that.

This absolutely sped up the development time, right? It also it also reduces the rework, right? Because you know what you're building at that point. You don't have to think about okay, when we bring this together, we know we're going to have to redo some things because it's not all going to fit. You iterate on it together throughout the development process. That makes the final product resonate more with what the business wanted. That's a real life example and I know that's very generic. Yeah, it's just bringing multiple teams together and making sure that everyone hears each other, right? Then they have a better understanding about what to develop together and they iterate collectively over the timeframe.

[0:16:33.6] JM: Describe some of the tools that have been helpful in the move towards DevOps.

[0:16:38.6] JJ: Absolutely. Tools, we've been very purposeful with the tools that we've selected here. One of the reasons that we wanted to spend a lot of time considering and mulling over exactly what we would have for developers here at Delta is because we want to be able to attract and retain top talent. For example, git is an absolutely necessary part of any DevOps, right? You have to know git, version for version control, or SVN, something of the sort.

We had a legacy version control tool. We decided to go with GitLab, just because of the ease of use, right? We wanted to have something that was easy to manage from an administrative perspective. For version control, that is what we decided upon. On the same note when it comes to developer recognition, Jenkins, that's something we also decided to leverage, because when you think of CI, you can't walk two steps without somebody saying Jenkins. That's another tool we decided to go with.

Other core offerings we decided to use are SonarQube for code quality and making sure that we were able to identify opportunities within the code quality space, as well as nexus for our artifact repository to ensure that we had some internal reference to all the dependencies that were being referenced from a Delta perspective. Those are the tools that we have currently in-house.

[0:18:05.3] JM: Those tools compose a pipeline through which the code moves from being in its raw form on the developer's machine to a shared hosted repository, to some testing

environment. Well, I guess before testing, there's a code review process. Then it goes into a testing environment, or perhaps even before then, it gets static analysis and I think you said Sonar.

Then you have static analysis and then it moves through static analysis and then you have a testing environment. Moves through a testing environment, then – or maybe multiple testing environments, perhaps you have some manual testing. Eventually makes it to production. Can you give me the overview of the code delivery lifecycle?

[0:18:51.2] JJ: Yeah. Let me just make sure I understand the question. You want to know exactly what it takes from that inception and idea to go out there into our customers' hands, right?

[0:19:02.6] JM: Yeah, yeah. Sure. Or I mean, to the extent that you want to talk about it to the extent you think it's relevant to DevOps workflow.

[0:19:09.0] JJ: Absolutely. We have implemented full CI at this point. We're still working on our continuous delivery strategy right now. I'll take it as far as CD, if you don't mind, since we're still forming up some of those items.

Basically, right now our process or our best practice for STLC is one, making sure that I'm going to even take it a little bit further back from the requirement, right? We use a term, or a tool version one to capture all of the requirements and translate them into meaningful user stories and making sure that we're collaborating with our business side in the form of a product owner engagement, before the development team even takes this into development. That's our first step. With that, we are also identifying any behaviors that we want to define and going from there.

As the developers get these requirements and work to understand them through the agile scrum methodologies and ceremonies, we then define our tests first and foremost, because we're driving test-driven development here at Delta, just to make sure that code quality is something that is always at the forefront.

Once we define the code quality aspects in the forms of different levels of tests, we're able to then start development. We have different types of development here, but mostly Java. We work with our GitLab version control, define a branching strategy for the team in which they can effectively promote certain versions and build them into their test environments for testing.

Right now, that's our process. Jenkins is what orchestrates that continuous integration. Right now, we're leveraging that for delivery into multiple environments as well. That's our process right now. We're working on defining our continuous delivery, which would of course help us define different types of deployments, canary and Bluegreen, different types of things, but we're still strategizing on that right now.

[0:21:08.0] JM: I've always been so bad at these terminology breakdowns. What's the difference between continuous integration and continuous delivery? Continues delivery is more holistic, or it's higher-level or something?

[0:21:19.3] JJ: Well yeah. Continuous integration would be the continuous building. I call it like just putting all the pieces together and creating an artifact, right? The end-state for CI would be an artifact, a binary that encompasses what the developers have developed. Jenkins is our chosen orchestrator for that. Nexus is where that artifact lives for us here at Delta.

Now for continuous delivery, I look at continuous delivery to be a standard methodology for delivering that artifact into multiple environments in different ways. Continuous delivery can facilitate Bluegreen deployments, which deploying a certain artifact in a subset of target servers and the routing traffic, their percentages across different environments. Right now, we are actually only doing continuous deployment, going to throw another one in there for you, which means that we're able to deploy that artifact to multiple number of target servers, but not able to continuously deliver it through an automated fashion. Three different terms can be pretty confusing.

[0:22:30.4] JM: I need flashcards. How does code review fit into this process and the collaboration set of things before it makes it into actually being built and static analysis and so on?

[0:22:46.0] JJ: Right. We actually are leveraging GitLab for our code review process right now. We don't have a specific tool, but GitLab has a lot of great features within their merge requests that allow you to collaborate on proposed changes. A lot of our teams are protecting the branch that goes into the CI process. Only if you complete a successful merge request are you able to then merge that code and generate that artifact.

Standards, naming conventions, all that good stuff, you have the opportunity to approve before those changes are merged into the target branch. That's how we are encouraging our development community to code review right now.

[0:23:26.0] JM: Do you use any cloud providers?

[0:23:28.0] JJ: We do. We are looking – I don't even know if I can say exactly whom. We're looking at multiple –

[0:23:33.1] JM: No problem.

[0:23:34.6] JJ: - cloud providers, public. We all so have an on-prem cloud as well. I think I can talk about our on-prem cloud, or what we're leveraging right now, which is OpenShift. We're looking at multiple public cloud providers to move into a hybrid cloud posture.

[0:23:50.9] JM: What's been useful about OpenShift?

[0:23:53.0] JJ: I think that OpenShift is a great tool in the orchestration of the Kubernetes and containers. I think it makes it easy for our large development and IT organization to adopt it. We've been able to create utilities and leverage a lot of the capabilities that OpenShift provides, such as routes, scaling within pods and all that good stuff, to make it easy for teams to adopt this. Because this is very new. You really want to create a little barrier to entry. Bare-bones Kubernetes is awesome, but OpenShift makes it a little bit more palatable to the masses.

[0:24:32.4] JM: Absolutely. I think that's the case with a lot of these – the newer infrastructure tools, or even the ways that older infrastructure companies are remaking themselves, because there's more and more companies that are needing to develop a pretty sophisticated software

development lifecycle. Just choosing the right set of tools for a company, let alone learning how to use those tools is quite difficult. It's like walking into a shopping mall, basically.

It's like, okay, you've been told you need a shirt, you need some pants, you need a hat and that's all you've been told, and a pair of shoes. You're like, “Well, but what kind of shoes? What kind of shirt? Is it a nice shirt? Is it a cheap shirt? Can I buy all this from the same store? Do they have to be the same color?” You have no idea how to mix and match these different things.

You could go to OpenShift and say, “Okay, OpenShift. I need this grocery list of things.” OpenShift will tell you, “Okay, well here's a list of things you can – we’ll give you all of this stuff. We’ll give you all the defaults. Here's everything you need.” You might say, “Well, but actually, this thing is going to break for me and that thing is going to break for me.” You actually need to have some mix of perhaps, going with some default solutions and some ala carte solutions. Can you tell me about the process of selecting vendors, of selecting tools and how Delta goes about doing that?

[0:26:05.2] JJ: Sure. Granted we are still in this process and we're still trying to figure out what is going to be valuable for Delta, what do these vendors provide that Delta could see some benefit from? I think that one of the things that we first need from a Delta standpoint is there has to be a need, right? Whether it's an application team that comes to one of the enterprise architecture representatives and says, “Hey, we would like to leverage this capability.” We start from there. All right, let's figure out this capability. Is there anything at Delta that we have right now that could serve the same need? If not, okay, let's look at this external capability that's provided by this vendor.

Then we understand okay, what value would leveraging this capability offer? Is there a way that we can use maybe something that's open source, or something that doesn't require – isn't required to pay for use, to get the same benefit? If not, okay, we take a look at if the value proposition is good for that business unit, or they just really, really need it to mitigate some risk, we go through the process of getting it proved through architecture.

We have, call them gatekeepers to make sure that we're not bringing in just any and everything into the Delta environment, because we don't want to duplicate any capabilities and pretty much

pay for tools that are already being leveraged here internally that we don't need to buy, just because – we just don't buy things just because. Once you get through that approval process, then it's the matter of figuring out is this going to be supported by the vendor? Because we have a very – our environment is very volatile, right? We have to react a lot, right? It's an airline.

We have to make sure that we have the support in order to maintain our SLAs from both external customers, meaning folks that get on the plane and also, our internal customers here, for example, development team. We have to make sure that whatever vendor we select has the ability to support an organization like us. Those are just a few of the things that we check out. We have of course, a formal process that we go through engaging with negotiating and that stuff. Really, it's about making sure that one, is what we're bringing in going to provide value to Delta? Two, can they support an organization of our size, from maintaining our SLA and making sure that we are successful with using the tool?

[0:28:44.1] JM: One tool that has risen in prominence over about the last four years is Slack. When I started Software Engineering Daily, we did a lot of shows about DevOps. That was about four years ago. Since then, Slack has become really pivotal in a lot of organizations. It's really altered the DevOps movement. Can you give me your perspective for how Slack fits into a DevOps workflow?

[0:29:14.2] JJ: Yeah, absolutely. I mean, I talked a little bit before about just collaboration and breaking down silos. Slack really does enable that. There's a few other collaboration tools to do as well. Slack is one of the leaders in this space. One of the ways that I think that it really shines is just the level of integration that you have. Being able to pretty much accumulate all of your meaningful data in one place from a development team standpoint, so that way the team can make informed decisions, I think is very, very valuable. As a development team, I'm able to know is my application is tanking via monitoring alerts, right?

Then you can have different type of, we call it ChatOps, if you will, I know that's another buzzword. You can actually make changes from your Slack channel to the application and fire off some automation to resolve, or go do a little bit deeper, get the logs, all that good stuff. I think it really helps that collaboration. Instead of a developer just getting an e-mail going about it

alone, digging into logs, you have the ability to make it a team effort, which is what DevOps is, right? It's about collaboration and breaking down those silos.

Because in the past, our operations folks would have troubleshooted based on development direction and tried to solve the problem on their own. Now you're bringing that altogether in one place, one location. Then you have that history for all time, so it makes documenting very, very easy. No one likes to document, but I think it's a little bit easier if you have that, all of those steps within one place. Slack does a great job with all of that.

[0:30:58.2] JM: I want to come back to the organizational changes and the behavioral changes, but I guess, just while we're on the subject of tools, could you just give me your high-level perspective for why tooling is important and how important tooling is to making this transformation? Because we both know that the organizational changes, the behavioral changes are really important, but my sense is that if you choose the right tools, they can reinforce certain behavioral changes that you want, so you can really get this positive feedback loop between your choice of tools and the behavioral changes. Maybe you could just give me your high-level perspective for how tools enable the right organizational changes.

[0:31:43.1] JJ: Yeah. I'm trying to think of an example here, because of course, we use a lot of the tools that a lot of organizations are using out there. Okay, here's an example. One example that I like to go to when I talk about okay, wrong tool, right tool, it's really – there's no one-size-fits-all tool for any organization. I think that one thing that groups should think about before selecting a tool is barrier of entry. Barrier of entry in a sense, how easy is it going to be for someone to get started?

Because the main thing that we aimed to do here at Delta was to get people started and develop a grassroots adoption movement. When you have low barrier of entry, it makes it a little bit easier for folks to get started on their own, because we all know that hey, maybe this project doesn't have funding to start changing tools. I've heard that many different times. When there's low barrier of entries and you make the value known, I think that teams have the easier time of adopting.

Once you get them in, right, you just start pumping them full of the DevOps good news, I call it, and letting them know that these are the behaviors that a DevOps team participates in from collaboration, making sure that awareness of code that's going in, branching strategies, organizing your work. Being able to facilitate all of that is very important. I think that a lot of the large players within the DevOps space do facilitate a lot of the basic necessities. If there's any specifics that an organization needs, I think it's important that they consider that.

For example, integrations, right? When you think about the larger DevOps pipeline, you want to make sure that everything plugs in, right? Because you want it to be a seamless flow and provide some continuity for that experience once the developers adopt it. Because then, it makes it easy when they get in for you to reinforce those behaviors and governance around if you have the masses there, right?

It's easy to get everybody to adopt those things that you all are educating them on. I would say, just make sure that low barrier to entries and ensure that the core, basic capabilities are present, which I think a lot of the tooling vendors out there already do that.

[0:34:07.8] JM: What are the organizational changes that have been hardest to make within the organization?

[0:34:14.9] JJ: I would say, okay, so I always structure it this way; you always have those very excited folks that are ready to just try something new, because the old way was not working. You have those early adopters and then you have the folks that are – they're interested, but you got to pull them in and say, “Hey, this is what's in it for you.” Then you have those folks that are like, “No. I don't think this is meaningful. I'm just going to continue doing things the way that we've always done them, because it works.”

I think that one of the hardest organizational changes really for anybody is just figuring out how to provide a value proposition to those folks that are kickers and screamers. They don't want to really adopt these new things, because it's very earth-shattering. A lot of these things that – a lot of these ideas that DevOps brings. Understanding and making sure that they – can understand exactly what benefits are in it for them, I think is very important.

What we've done is we have a lot of folks that have been at Delta for a while, right? Just getting those champions, right? In the legacy space and making sure that hey, we understand their perspective, right? We can really tune and say that, “Hey. Well, I know that you face this issue within a legacy tool that you're using. Well, the great thing about DevOps is this is how it solves that,” right? Really, just addressing that specific thing and then okay, all right, well let's see what this DevOps thing is about then.

I think that's one of the challenges is getting those folks that are really gung-ho and those behaviors and processes over to the party and making sure that we convey and can speak to what is valuable in them, in order to get them to transition over and adopt some of these DevOps methodologies.

[0:36:09.2] JM: Delta, there was the move towards DevOps involved a dojo. Can you explain what a dojo is?

[0:36:16.8] JJ: Absolutely. Got this idea, I don't know if you're familiar with the Target Dojo at all, but two years ago, I think now, we had a visit to the Target Dojo. The whole goal of it was to just bring together development teams and the business side and collaborate and build awesome things.

We built one here in Atlanta. Essentially, our dojo, it's called the speed to market dojo. It's a place of immersive learning and that's what we positioned it as here at Delta. Delta's dojo, we bring together different areas of excellence. We have API, design help, we also have DevOps, QA and quality engineering as well, cloud and security. I mean, all of those technical pieces are brought together through agile coaching. Pretty much those six coaching competencies are positioned in the dojo, and we immerse teams in learning on how you can adopt these new ways of working, in order to deliver value faster, better, with more quality, right? That's what our dojo is here at Delta.

We've had over, I think about 27 teams and we've only had it open for about a year and a quarter right now. It's been very, very impactful, I think when you think about adoption and transforming an organization. It's very easy to educate, right? We get our training materials and we go out and we teach.

Having an immersive learning experience is absolutely necessary, because when you can teach someone a new method while they're actually doing their day-to-day work, it resonates, I think 50%, if not more, I mean, better to them, because they can actually see how it makes – it applies to their day-to-day life, versus just conceptual knowledge. Yeah, that's our dojo and it really has helped with adoption of a lot of the best practices across our competency areas.

[0:38:21.0] JM: What are their learnings have you had talking to other companies, other large and established enterprises that have gone through a DevOps transformation?

[0:38:35.0] JJ: I think that the target is one that comes to mind, I know that we have to focus on bringing the business closer, right? Making sure that at the inception of a lot of these development efforts that everything is clear and you have close collaboration with the business. That message has been very, very clear through target. They do a product owner training. They encourage the business to participate in a lot of the scrum ceremonies. That has helped make the development process and life cycle a lot better and smoother, because you don't have any just misinterpretation, right? You have everything clear from the forefront. I think that's one of the learnings that we've gained so far.

[0:39:21.8] JM: What advice would you give from your own experience about in what other companies could take and undergoing their own digital transformation?

[0:39:32.5] JJ: I would say, one piece of advice is figure out what works for your individual organization. I think that a lot of companies have stories. Just because one company did and it worked for them, doesn't mean that it'll work for your specific company. I think that a lot of the stories we hear are awesome, but each company has their own unique problems, right?

It's great to start there, but figure out what unique problems you have within your space. Then work on addressing them within either tech – not from technology, maybe it's a people thing, or maybe it's a process thing. Understanding those specific issues is an absolute must and not just taking a blueprint from another company, but work and examine your own internal structure, your organization, to figure out what unique issues do you have and then work on addressing those as well.

[0:40:30.6] JM: You're going to be speaking at the GitLab Commit Conference. What's your talk going to be about?

[0:40:37.0] JJ: Yeah. Our talk – I have two talks. The one that I'm doing solo is about avoiding the vendor law when you're looking at implementing cloud native applications, whether it's a public cloud, or on-prem. How can you strategically tool your organization, so that you do not become dependent upon a vendor? Because of course, there's a lot of competition out there in the public cloud world. How can you ensure that you maintain that flexibility?

I'm doing a joint talk with Chris Bolton on my team. He's an awesome engineer. We're going to be talking about how Delta is a marketing planar cloud native journey. I won't give too much away, but we'll talk about what we're doing, maybe what public clouds were planning on leveraging and how we plan on pretty much building out our processes, in order to facilitate adoption of all of this new stuff.

[0:41:33.0] JM: Do you go to a lot of these tech conferences? What value do you get out of going to tech conferences and communicating with people at them?

[0:41:40.5] JJ: Absolutely. These are the highlight of my year. The reason being is because I get an opportunity to talk to those folks that are on the ground implementing these things. One great, one I went to going back to the dojo was a dojo consortium. It pretty much was a collective of all of, I don't know, is dojo is a word, but the dojos across the US that had built dojos, or were in the process of building them, let me have the opportunity to say, "Hey, this is how we solved our problems. Go ahead and you try it this way."

Getting the opportunity to get your direct questions for your direct issues discussed with people that have been there and done that, I think is absolutely meaningful, because sometimes you flounder a little bit when trying to solve these problems. If there are folks out there who have done this, why not just go straight to them and see exactly what worked for them? I love it.

[0:42:33.0] JM: All right. Jasmine James, thank you for coming on the show. It's been really fun talking to you.

[0:42:36.8] JJ: Thank you so much for having me, Jeff. Really, really appreciate it. It's been great.

[END]