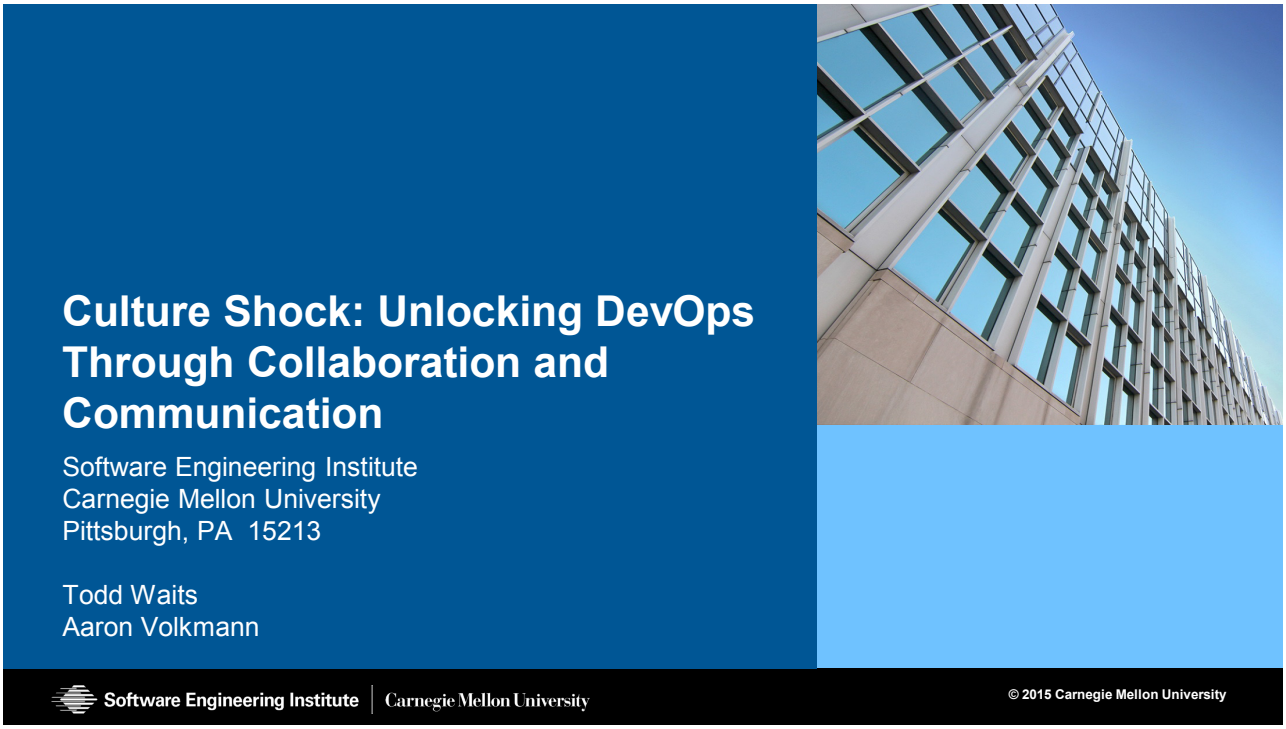
Culture Shock

Table of Contents

Carnegie Mellon University

**001 01_T01 - Culture Shock

Live Stream.pptx



**003 Shane McGraw: And hello from the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania. We welcome you to the Software Engineering Institute's webinar series. Our presentation today is Culture Shock: Unlocking DevOps through Collaboration and Communication. Depending on your location, we wish you a good morning, a good afternoon, or good evening.

My name is Shane McGraw. I'll be your moderator for the presentation. And I'd like to thank you for attending. We want to make today as interactive as possible. So, we will address questions throughout the presentation and again at the end of the presentation. You can submit your questions to our event staff at

any time by using the questions tab on your control panel.

We will also ask a few polling questions throughout the presentation. They will appear as a pop-up window on your screen. The first polling question we like to ask is how did you hear of today's event.

Another three tabs I'd like to point out are the files, Twitter, and survey tabs. The files tab has a PDF copy of the presentation slides there now along with other DevOps related work from the SEI. For those of you using Twitter, be sure to follow @SEInews and use the hashtag seiwebinar. The file tab you can open up by the end of the presentation. And we hope you fill out as your feedback is always greatly appreciated.

Now, I'd like to introduce our presenters for today. Todd Waits is a project lead at the SEI with a background in entertainment industries and entrepreneurship. Todd provides a unique perspective to help improve workflows and processes with innovative technologies solutions. He helps law enforcement and government agencies embrace modern technologies and development methodologies.

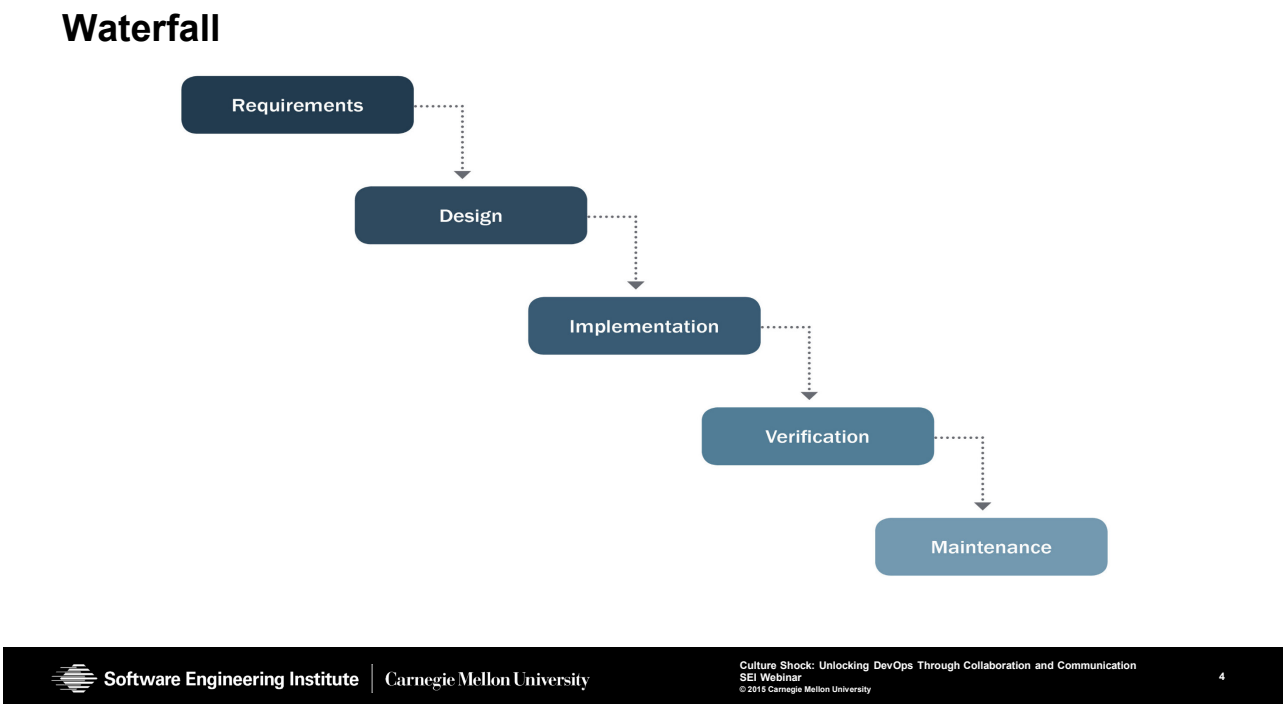
Aaron Volkmann is a senior research engineer within the CERT division at the SEI. He specializes in product and research driven development. Since 2003, he's helped numerous

organizations in the government, as well as healthcare, retail, software, and manufacturing industries meet their business objectives through technology. He holds a masters in computer information systems from Boston University.

And now, I'd like to turn it over to Todd Waits. Todd, welcome, all yours.

Todd Waits: Thank you so much, Shane. We're really excited to be able to be able to speak to you guys today about DevOps and culture, collaboration, and communication issues. Before we get started going into the communication collaboration piece, we wanted to give a brief overview of where DevOps came from to give an idea of why there may be some of these cultural issues that we're facing.

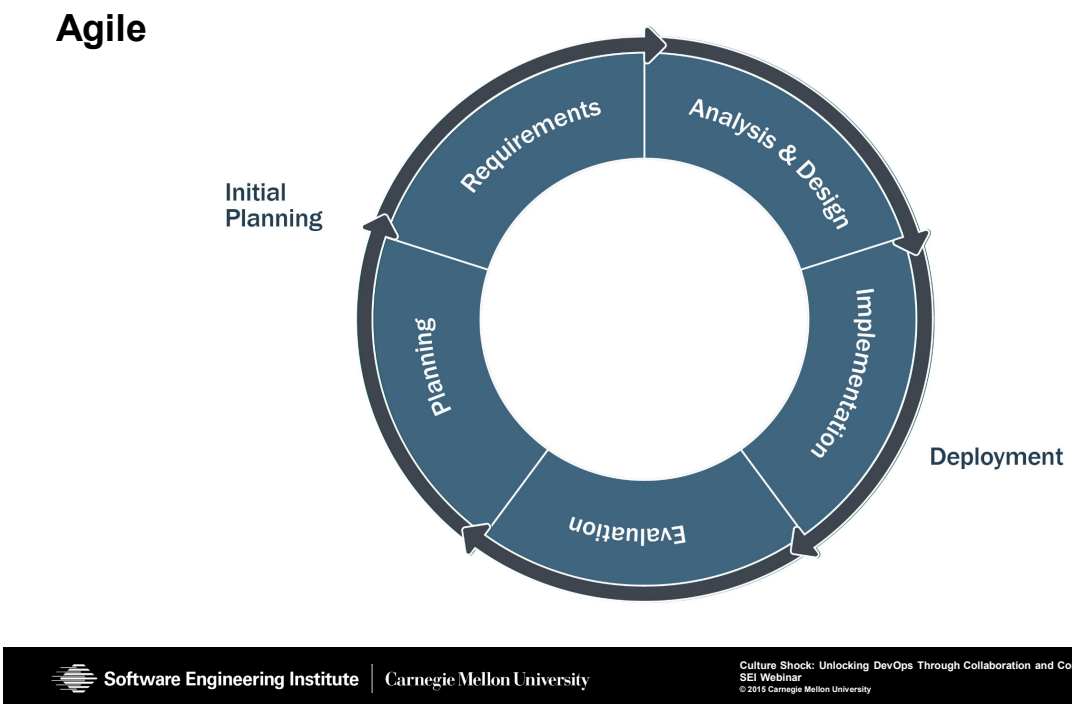
Waterfall



**004 So, to get started, we want to talk about waterfall. So, waterfall methodology was kind of the original software methodology. It mapped really well to the business models that people were using. You had very clear functions, and timelines, and roles. And one led right into the other.

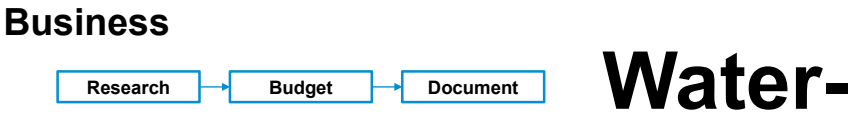
However, that ran into problems as requirements would change. And it's a complicated system with a lot of moving parts.

Agile



**005 So, we started using iterative cycles, the most prominent of that being Agile. Agile was around to bring the customer closer into the development process so that we could focus on building what customers actually need. So, we made sure key stakeholders are a part of that iterative process to generate a product that everyone could use.

Business

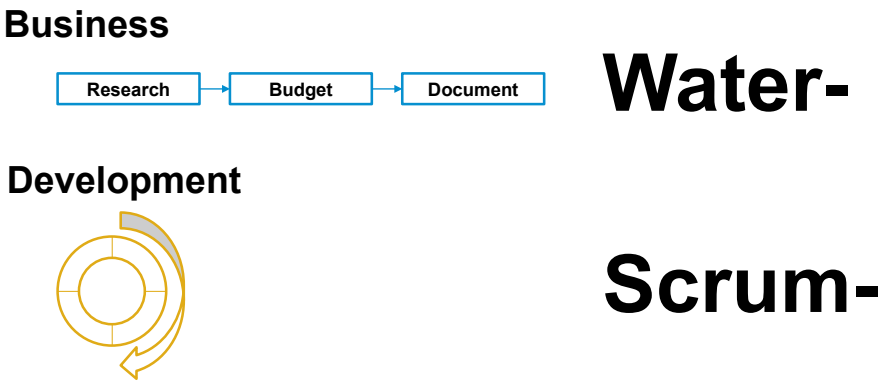


Jez Humble, https://youtu.be/L1w2_AY82WY

Dave West, http://sdtimes.com/analyst-watch-water-scrum-fall-is-the-reality-of-agile/

**006 Aaron Volkmann: So, as enterprises started integrating Agile, what we started seeing in reality is--

Business

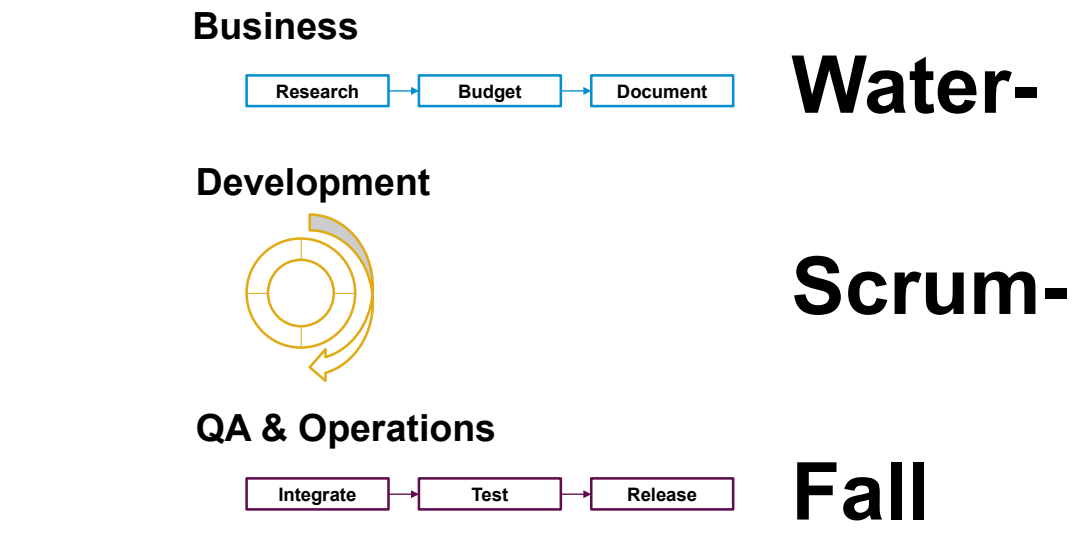


Jez Humble, https://youtu.be/L1w2_AY82WY

Dave West, http://sdtimes.com/analyst-watch-water-scrum-fall-is-the-reality-of-agile/

**007 Water, scrum--

Business



Jez Humble, https://youtu.be/L1w2_AY82WY

Dave West, http://sdtimes.com/analyst-watch-water-scrum-fall-is-the-reality-of-agile/

**008 Fall, where we have development in the center aerating, doing their daily standups and releasing vertical slices of functionality into production. But at the top, we have the business who's forced to plan out everything, forecast the whole fiscal year in advance, how much money they're going to spend, how many team members the must have. And then at the bottom, we have QA and operations with a centralized quality assurance team that must integrate everything together with all the existing systems in production, test everything out, and then transition it to operations so that it can be released into production. So, there is sort of a disconnect here.

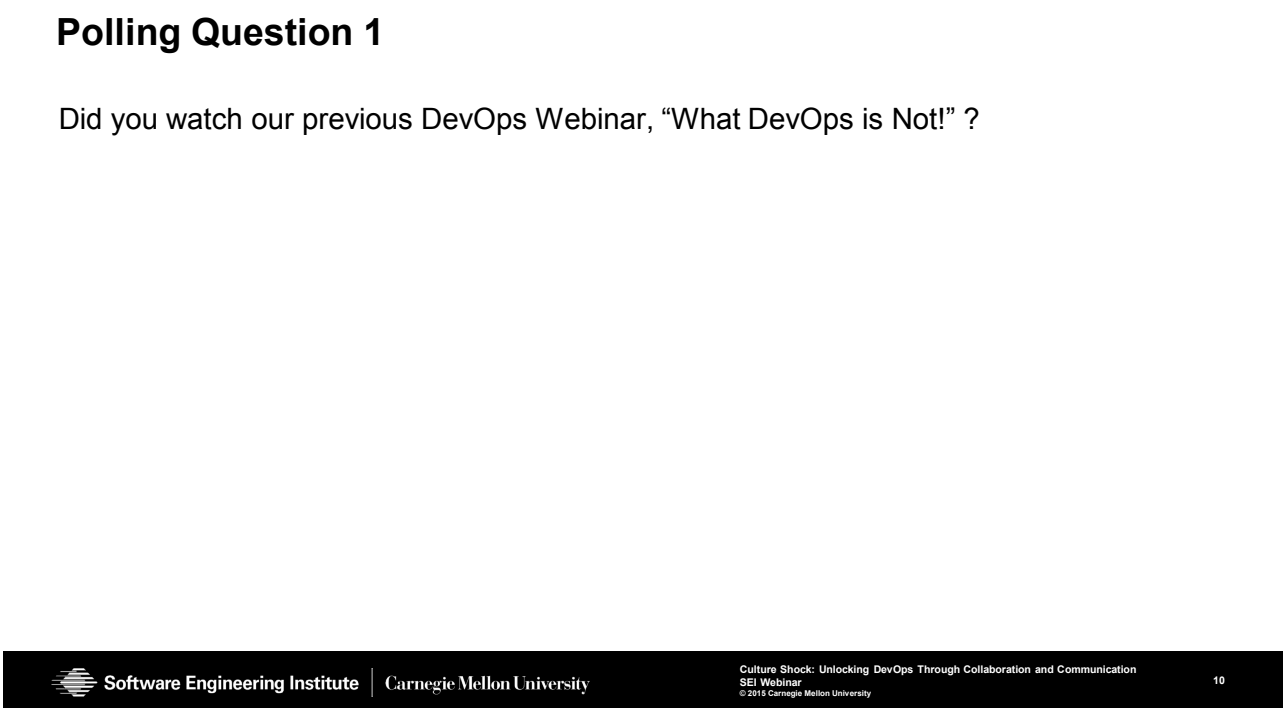
DevOps

**009 And that's where DevOps comes in. Sort of like how Agile sought to bring the customer closer to the development process to assure that we're developing what the customer really needs on a schedule that they need, DevOps seeks to align development and operations. So, through aligning developments, and operations, and security along business needs through shared goals, giving them all the shared goals through collaboration, we have DevOps that seeks to accelerate developing software with quality as quickly as possible.

Todd Waits: I think one of the foundations that I really like about the DevOps idea is the collaboration piece. Really, if you're just getting started trying to implement DevOps,

whether you're a large shop or a small shop, the key point here is the communication. Initially, it may start with just opening your mouth and communicating with another side of the fence there talking between dev and operations making sure everybody's on the same page.

Polling Question 1



**010 Shane McGraw: Okay. So, we're going to jump in ask-- like I mentioned during the intro, we're going to ask some polling question throughout the presentation to kind of help us drive the flow of the presentation. So, the first one we're going to pose here is did you watch our previous DevOps webinar by Hasan Yasar and Aaron Cois. And that was titled What DevOps is Not. That will give us an idea of who was

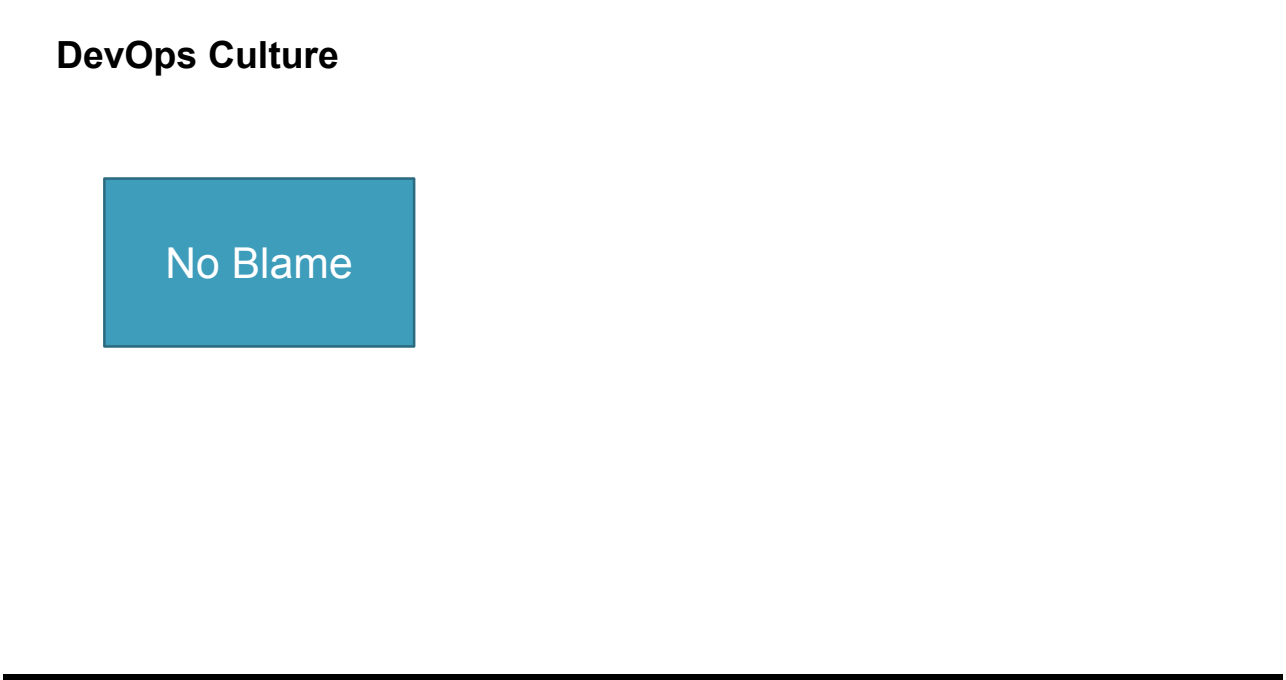
in the audience. We'll give you about fifteen seconds or so to vote there.

And while you guys are voting, I wanted to let everyone know based on some feedback we got from the last webinar, we decided to start a DevOps forum on LinkedIn. So, we invite you all to join that forum if you have a LinkedIn profile, just got o LinkedIn. Within groups, you can search on DevOps. And you'll see that forum. And we invite you to keep the conversation going after the webinar there.

So, we will show the results here for our question. We had seventy-four percent no, and twenty-six percent yes. So, you may have to cover some of the basics again. Todd, so back to you.

Todd Waits: Excellent. Well, thank you all for joining. Those who didn't get to catch the last webinar, thanks for joining us today.

DevOps Culture

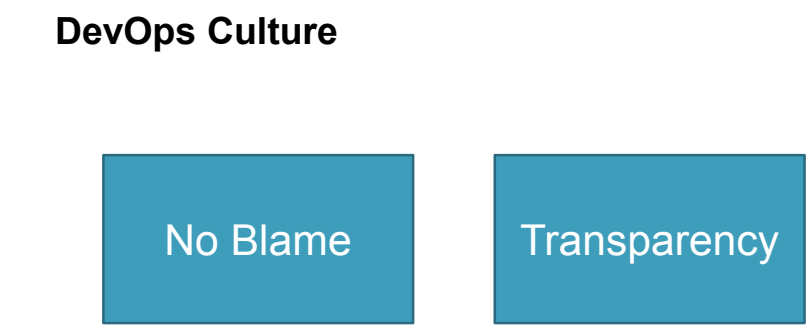


**011 Aaron Volkmann: So, next we're going to describe what we feel like are some of the attributes of a DevOps culture. First, it's a no blame culture. We have no time or energy to have blame and emotion brought. We have to focus on getting the job done. Things always go wrong. Whenever they go wrong, we have to focus on identifying on fixing the problem and identifying what went wrong so that it does not happen again.

Part of a no blame culture is also creating a culture where we're willing to take risks. And in fact, we get close to the edge of risk taking with prober risk mitigation so that we can properly innovate and push the envelope of what we're doing with technology. If we're in a blame

culture, then technologists are going to be afraid to take risks and create good innovation.

DevOps Culture

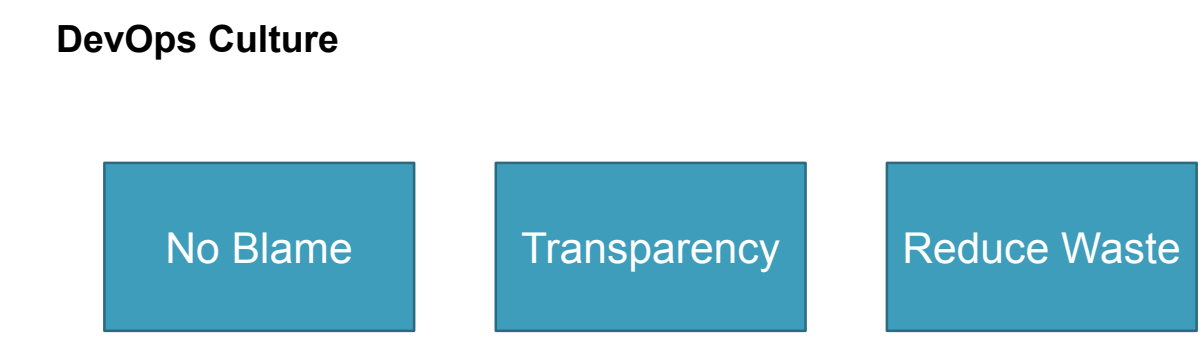


**012 Todd Waits: That's right. In order to be able to take proper risk, we also then have to have proper transparency for the organization. That is one thing that helps in avoiding a blame culture is providing as much information and transparency as possible into the process and into the artifacts themselves. That transparency-- ways to achieve that are using tooling such as issue trackers and documentation. And it can be as basic as opening your mouth and talking with other people, making sure that the necessary meetings are occurring. And that there's awareness

through the use of automated tooling and whatnot that can present the results and present a picture, a well-defined picture of the current status of the project.

Aaron Volkmann: Right, empire building within an organization helps no one but whoever's building an empire is to the detriment of the organization overall.

DevOps Culture



**013 Another characteristic is a passion for waste reduction. Just like Toyota production systems identifying seven areas of waste, you can check out our LinkedIn group to learn more about this after the presentation. But what I'd like to just hit on is that overproduction where maybe we released too many

features than what the customer can consume. This is a type of waste because if we release things that the customer's not going to use, we don't know if by the time they're able to use it, if that's really what they need at the time. So, that's a very big type of waste.

Another type of waste is inventory where we have too many works in progress sitting around that represent risk once they're started. If they're just sitting around, they represent risk due to QA costs, the cost of starting those initiatives up again, and the changes if they're left to sit on the shelf for a long period of time.

Perhaps, in DevOps the biggest source of waste is unused creativity. If you're entire team is consumed with putting out fires all the time, then they aren't going to have time to use their creativity in order to improve processes that lead to better outcomes.

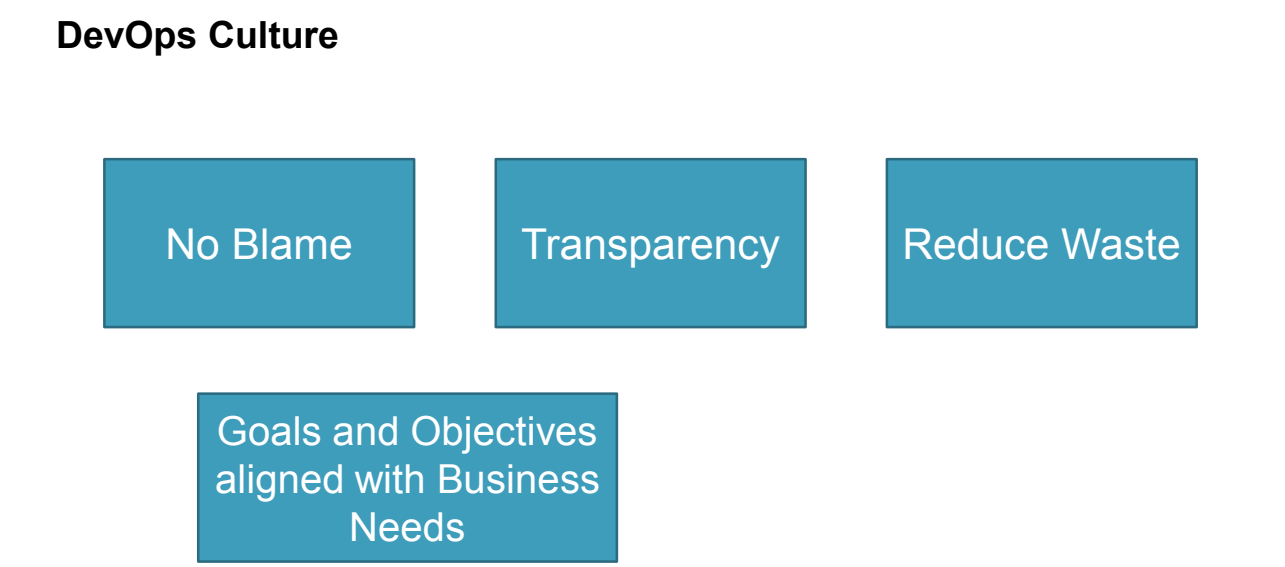
Todd Waits: I think a lot of times you'll find that the people that are closest to that work, while they are putting out those fires, they're also identifying a lot of ways that they can resolve those fires so that those problems don't occur in the future. Sometimes we, as managers or leads, will look at these problems and try to institute what we think is the best solution. And it's important to take a step back and realize that we're here to support those people that are actually doing the work. And

let them recommend and guide what potential solutions will be.

Aaron Volkmann: Right. Fostering a non-toxic environment where everybody's comfortable sending up their feedback on how to improve processes.

Todd Waits: Absolutely.

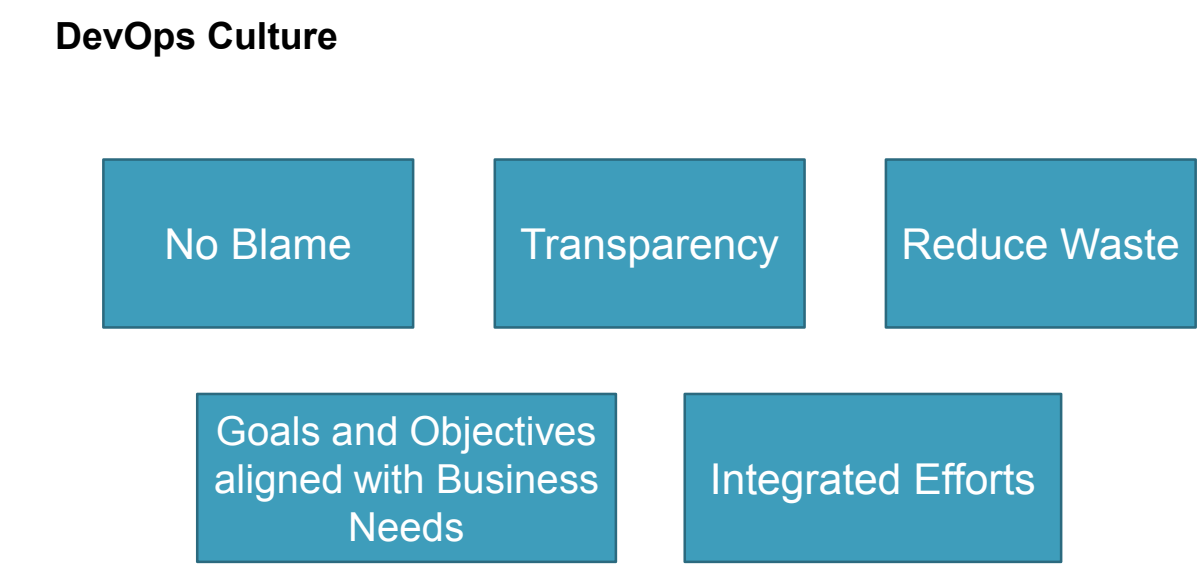
DevOps Culture



**014 Aaron Volkmann: Another aspect is making sure that goals and objectives are aligned with business needs. This very closely related to org structure, where the organization has non-conflicting priorities. This goes with pooling development and operation resources to make sure that in order to get a job done, everybody is tasked appropriately to

be able to get that job done on time. I think this carries over to making sure that business needs are aligned. And businesses properly utilize their IT organization so that a manager over here and a manager over there aren't competing for IT resources that could throw an initiative off track on the IT side of the house.

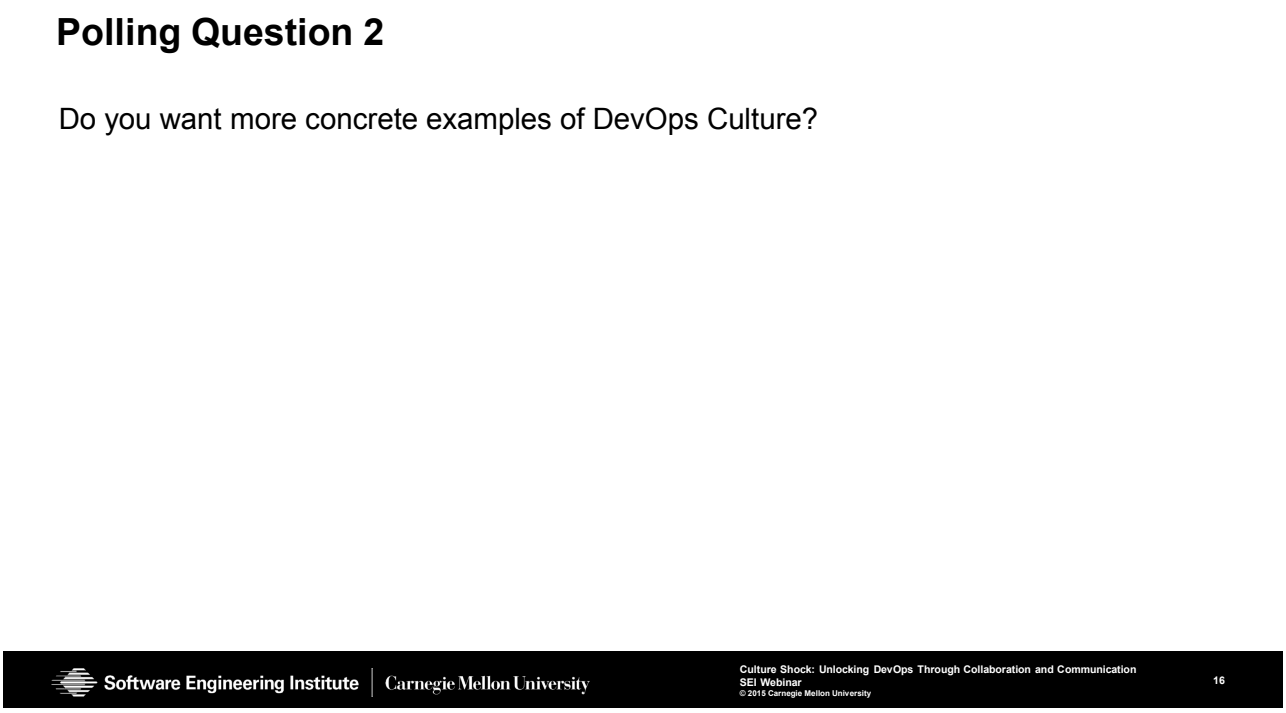
DevOps Culture



**015 Todd Waits: I think an extension of aligning the goals and objectives is the idea of integrating efforts. So, as Aaron mentioned before, where Agile was bringing the customer into the development process, DevOps is focused on bringing operations and all the pertinent individuals that have the information needed to deliver business value into the process. So,

we need to work to ensure that all those people that are needed are integrated and feel as part of a team and not isolated siloes spread about the organization. But it should have those aligned efforts and also are integrated, know each other, and are working together as a team.

Polling Question 2



**016 Shane McGraw: Okay. So, this is going to lead us to our third polling question. And that question is do you want more concrete examples of DevOps culture. Again, do you want more concrete examples of DevOps culture? So, we'll give you about fifteen or twenty seconds to vote there.

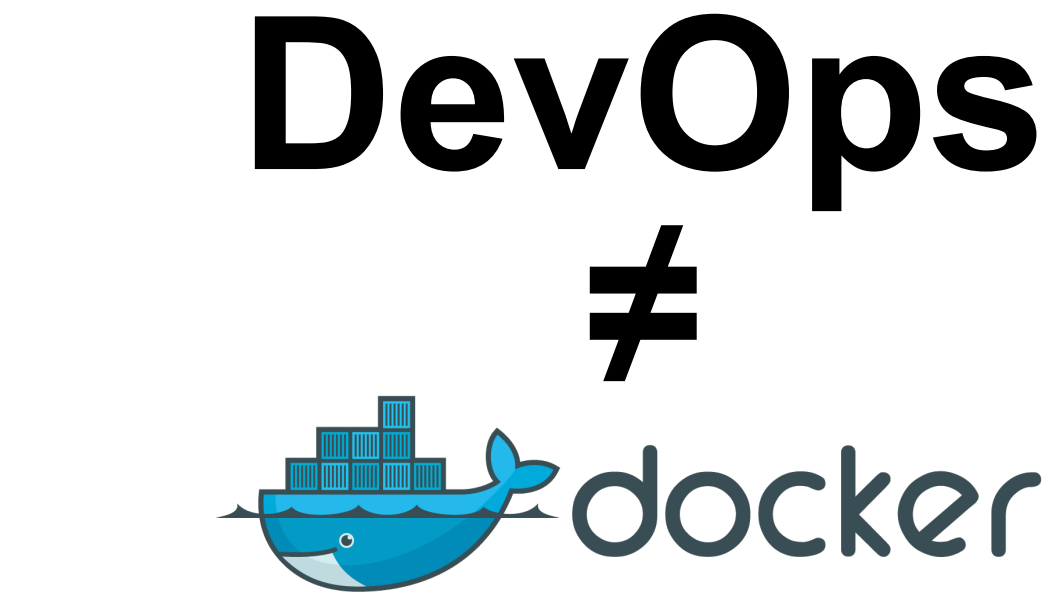
While you're doing that we got a couple questions come in from our

last polling question where we asked if you had watched the last DevOps webinar or where to find that. The easiest thing to do is you can just email info@sei.cmu.edu. We'll send you the location. You can watch the archive of that recording. Also, a couple questions on where to get today's presentation slides. In case you missed the intro, they are available now in the files tab. You can walk away with that and other resources related to DevOps today. And of course this presentation is being archived. And we'll send out an email at the end of-- most likely by tomorrow morning with the location for that.

So, let's share the results from this one. And we got eighty-six percent saying yes, they want more concrete examples. So, there's your challenge. Can we work more examples in throughout the presentation?

Todd Waits: Yes. I think we'll try to focus on getting some more concrete examples. That's excellent. Thank you. Also, I just wanted to mention that we can put some of these links into our LinkedIn group so that those who are participating in the discussion on LinkedIn can have access to those resources as well.

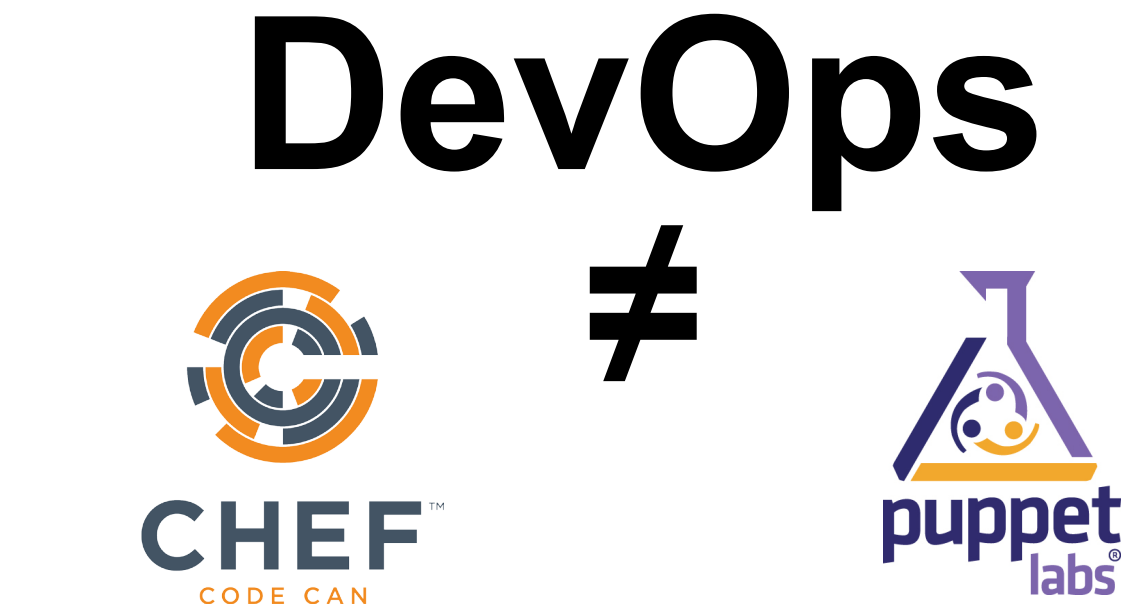
DevOps



**017 Todd Waits: All right.

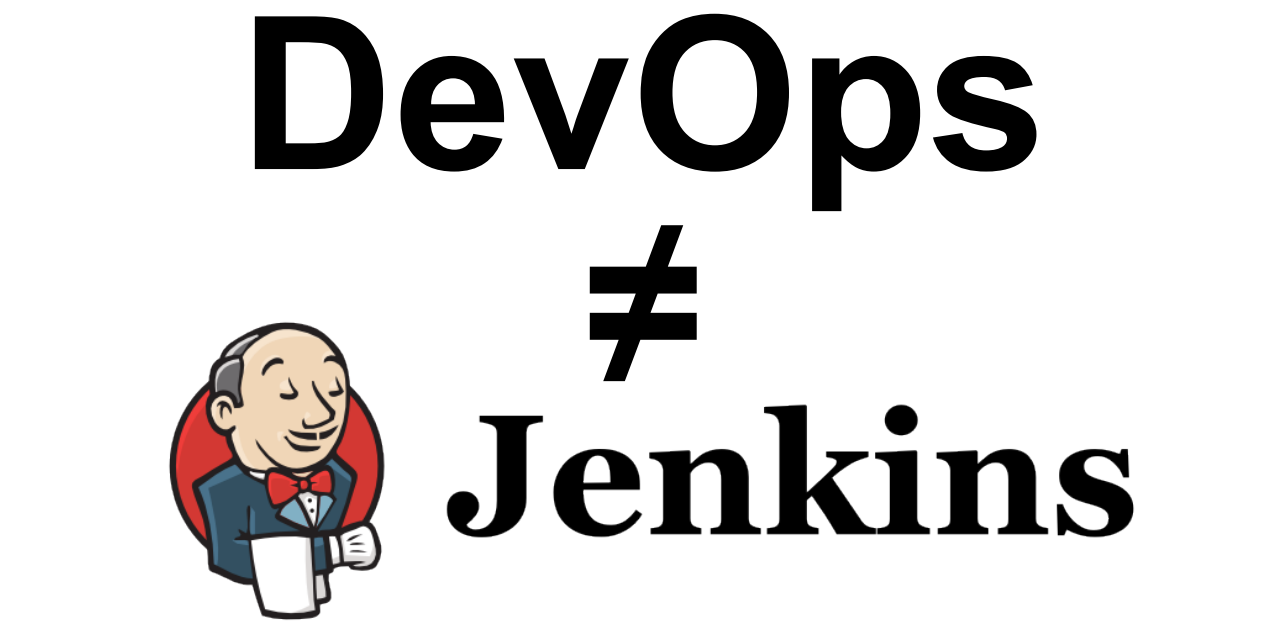
Aaron Volkmann: So, we'd like to talk about what DevOps is not since this is all about communication and collaboration. So, DevOps is not just containers.

DevOps



**018 DevOps is not just infrastructure as code.

DevOps



**019 DevOps is not just continuous integration, continuous delivery.

DevOps

**020 DevOps is people.

Todd Waits: A lot of times as we talk about DevOps, the tooling is kind of, it is a cornerstone of DevOps. But sometimes we can get so focused on what tools are we using that we often forget that without the people, DevOps really falls apart. So, I find myself feeling a little bit like Charlton Heston in "Soylent Green" saying, "DevOps is people," and feeling like a madman at time.

Aaron Volkmann: Yeah, sometimes the tools are a lot more fun than the people to deal with. So, it's very tempting to get caught up in the tooling.

Todd Waits: Yeah, as far as DevOps being people, what's important to

remember here is that with-- that it's about the communication. That being able to work-- if developers are working in an isolated silo, and they package something up, and they pass that on to the operations team, there's this vacuum. Or there's just this artifact that's passed back and forth. And so, what that can generally do is create grumble pits at best in the sense that once operations gets this packaged piece, they get all frustrated because the developers didn't implement something right or oh man, they just don't know what they're doing.

And the converse is true. When developers get information from operations they just say, "Ugh, these people just-- they don't understand what we're going through and what we have to do."

Aaron Volkmann: "Send it back. Send it back. We can't take this. We can't run this."

Todd Waits: Exactly.

Aaron Volkmann: Causing developers to disappoint their business customers, missing deadlines, things like that. So, a lot of animosity can form between these groups especially when they don't work closely together.

Todd Waits: Right. That's one of those issues with misaligned business goals and objectives. A developer's goal is to introduce change and new features. And the operations goal is

to maintain stasis and keep the applications running smoothly.

There was a study done back-- at McGill University in Montreal by Professor Jeffrey Mogil. And this experiment was to measure stress and empathy that people have toward strangers and also toward people that they know well. And what was found, which isn't really surprising, but is nice to have documented in a research environment is that we have a great deal of empathy toward people that we know, that we can see face to face, that we can interact with.

Aaron Volkmann: And people that don't stress you out.

Todd Waits: Exactly. So, when you introduce those competing goals and objectives, that introduces stress, which will then decrease empathy that we have between people. And we often find ourselves blaming the other group for the shortcomings of whatever process. And so, what the study found was that if you just introduced fifteen minutes of a shared experience-- and in this experiment, they used Rock Band. And they had strangers play Rock Band together for fifteen minutes and then proceeded to do this experiment. After the experiment-- the results of the experiment said that the strangers, after they played Rock Band they exhibited almost the same amount of empathy that people who knew each other previously

exhibited because they were able to share that experience.

Aaron Volkmann: Right. And so, getting people familiar with each other, and anything to do from-- a colleague here at the SEI was quipping whenever we're sharing this research was they tried lunches, that didn't work. They tried pizza parties, that didn't work. They tried after work socials to try to get different teams together, that didn't work. What they ended up playing was network Doom Death Match. And that ended up bringing the teams closer together and was successful. So, I think that's along the same vein.

Todd Waits: Yeah, introducing some non-stressful areas to just going to lunch or sometimes a lot of what happens is people just devolve into pre-existing cliques. So, your operations people will come to lunch with the developers. And they all sit-- the developers sit with the developers. And operations sit with operations. So, we don't want to do things that are going to further engender these cliques. We want to do things that will present low-stress activities that will increase empathy and interaction between one another.

I find a lot of times it happens is I'll submit a request to IT, and I'm much more empathetic and responsive once I've actually met the person that's fulfilling that request. It becomes a lot less ugh, these people don't respond and more okay, we

understand where they're coming from.

Aaron Volkmann: Right. Right.

Shane McGraw: Can we work in an audience questions related to this topic here now.

Todd Waits: Okay.

Shane McGraw: Damian wants to know what are some organizational barriers that you have seen impeding DevOps from coming into an organization.

Todd Waits: That's a good question. I think we're going to hit on that when we talk about some organizational structure. But I think we can just mention that right now is that organizational structure can be broken down in ways that can inhibit even the ability to gainfully discuss some of these issues that may be because of those competing objectives. I don't know if we want to go into more detail now or--

Aaron Volkmann: Yeah, I think competing priorities, groups wanting to do things the way they always have, that sort of thing. Whenever org structure prevents that from happening, whenever there isn't just one leader sort of encouraging or fostering an environment where everybody can do things using a new philosophy.

Shane McGraw: Okay. Back to you guys.

Todd Waits: All right. So, we have all this talk about DevOps is people. But we do want to make it very clear that--

People use tools

**021 People use tools.

Aaron Volkmann: Yeah, let's face it. We've got to use the tools. Take warning organizations out there. If you don't use some of these tools, then you're very far behind. And I don't know if I'd want to work with you. I'd work with you to implement some of these tools. Just like they say money can't buy you happiness, well it sure helps. And the tools by themselves aren't going to give you-- help you implement DevOps, but they sure help a lot. If you don't use tools like this, you're going to have

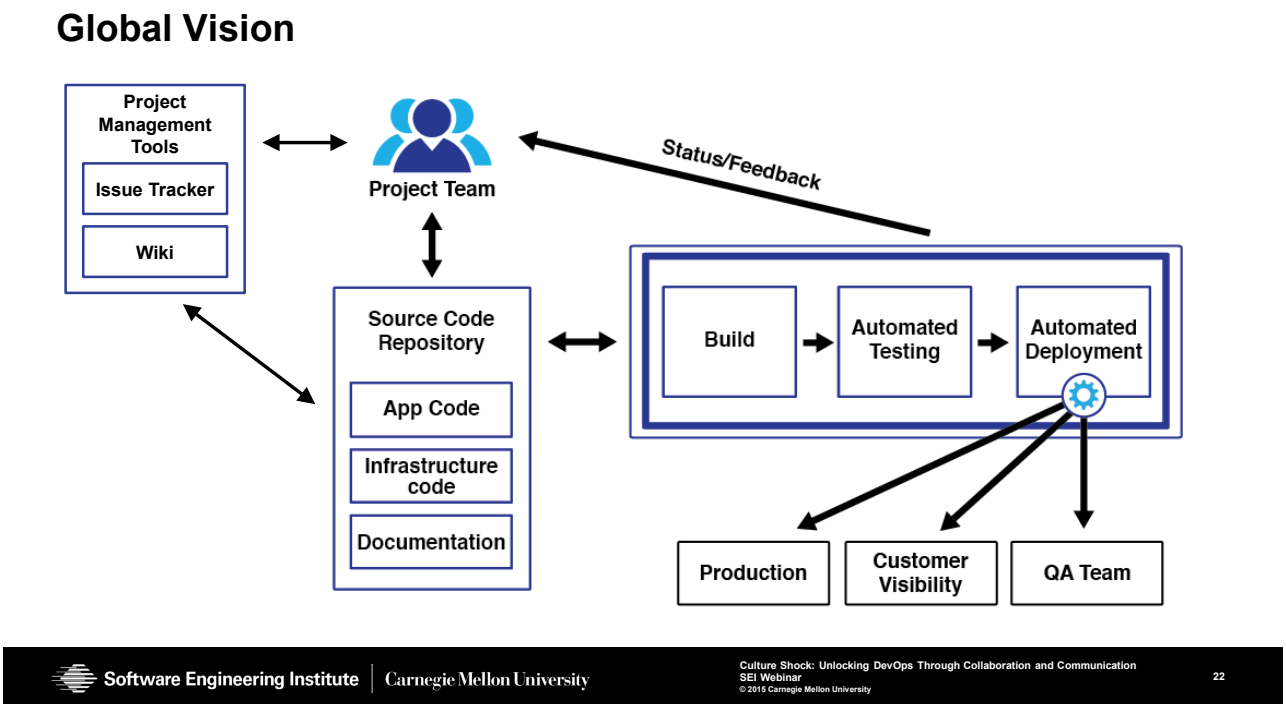
suboptimal results at best because DevOps is all about automation.

Todd Waits: I think one thing we kind of failed to mention in our previous discussion when we were talking about people is this idea that even-- you can start down the path of implementing DevOps without starting to implement these tools right away. And that is by opening up that communication by sitting down and saying here's some of the problems that we're facing as an organization to meet our business needs and starting the conversation to see how that can work. And as you go about increasing the collaboration, we can introduce these tools to a greater degree to greater effect in order to get DevOps working.

Aaron Volkmann: So, a one man shop or a small startup where everybody's sitting in the same room, they, by their very nature, are DevOps. But they would-- they're getting suboptimal results if they're not using one of these automation tools.

So, the tool and what it gets us, again, by itself--

Global Vision



**022 Not as important. But when used in a DevOps context, it gives us this global vision. And so, where we have a repository of the state of the application, kind of this persistent history of where the application came from, how did it get started. We have insight into what's the current state of the application. And even through our issue trackers, our Wiki, we can project into the future to see what will our application be in the future. And so, it gives this kind of three-sixty vision of where we're at with the application.

The automated testing, the builds, they give us the instant feedback and notifications of what's going on so that people can react and respond in an on demand sort of way.

Aaron Volkmann: I think another thing that related to global vision is through using all this automation, you can have all the data stored about your processes stored in a data store to enable easy auditing. You can go back and look in the past and know exactly what happened when. So, any organizations that must maintain SOCS or PCI compliance, this is right up your alley to use automation tools that store this data.

They also, just to touch on a couple of points, they free up humans' time so that we're not doing things manually and we have more time to innovate. They reduce error because we have a well-oiled machine performing a task hundreds of times maybe over and over again. And they just speed up everything, which is what one of our goals is is to speed delivery of quality software into production where it can give business value.

Todd Waits: For example, if you're deploying to your staging environment once every quarter to do your test, then you have that number of times to determine how successful you're going to be when you go to production.

Aaron Volkmann: The whole muscle memory metaphor.

Todd Waits: If you're deploying a thousand times a week to your staging environment, well then you're going to have that much more confidence that when you want to go

to production, it's going to work because we have that historical record saying that we've worked nine hundred and ninety-five times out of a thousand. We're going to work when we get into production.

The other thing that we introduce here that we put into this global vision is our documentation. A lot of people are documenting their source code. But we took this a step forward because, again, going back to the comment that we made earlier about the people closest to the work being able to be the ones who drive change process.
One of the things we noticed as we were working with our customers is that our requirements documentation and our design documentation was constantly sore spot. We spent more of our time finding the right version of a document, correcting issues because nobody had the latest--somebody lost the last updates. And so, we had a poor document management system going on.

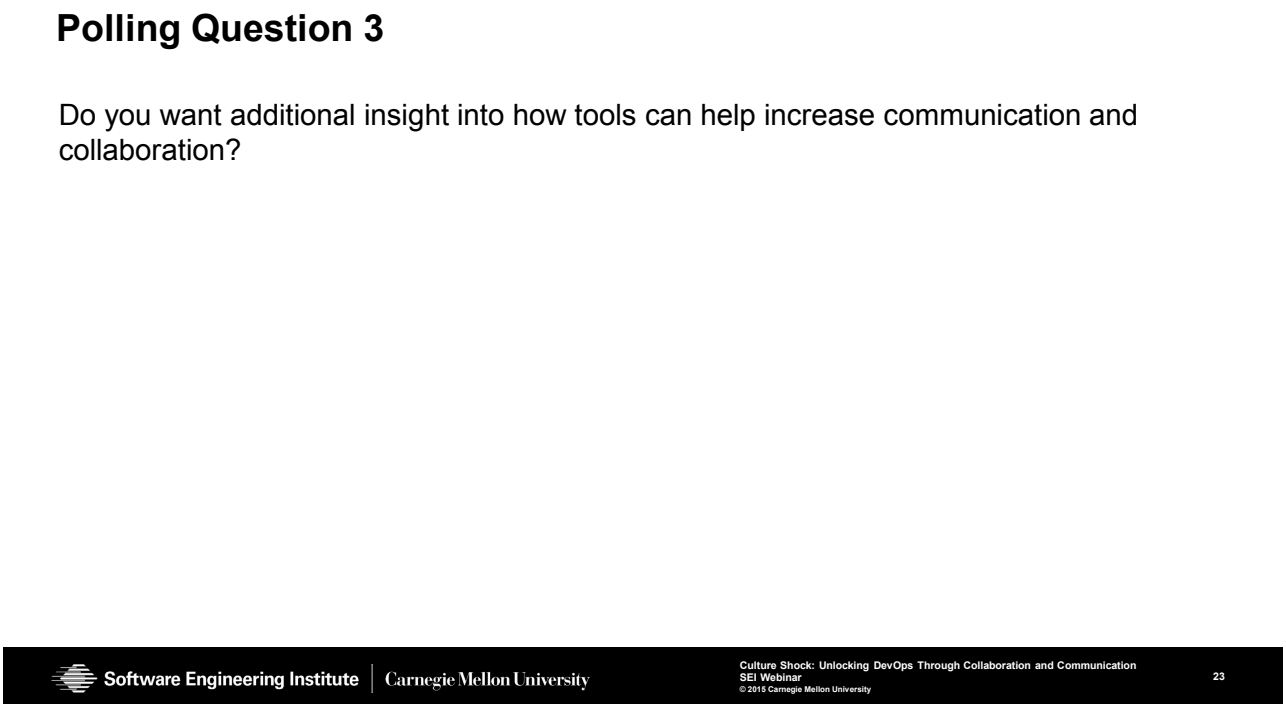
What we decided to do and what worked for our team was to put our documentation into our source code repositories. By putting it into our source code repositories, we can rely on the same rigorous testing that our code goes through. And so, our documents are checked in and built by our build server and then deployed into an Internet site that is then exposed to managers, stakeholders, customers. And they're able to always access the most current version. And we're even able

to tag build numbers with specific numbers of documents so we can always roll back and go to specific versions and see what the history of that document was so we can find out how did we get here.

So, it's the idea that as you're working in this, you as your team can identify areas that are important for you to put into this idea of what needs to be communicated and transparent for all the stakeholders involved. And you may start with just one thing. It may just be a build server. And then you may be doing your testing. You can start small and build on that as it works for the organization.

Aaron Volkmann: Right. And instead of having people respond to help tickets and put out fires all the time, they're working on building capabilities that then just give value and economies of scale as time goes on. And that's the kind of thing that we like to focus on.

Polling Question 3



**023 Shane McGraw: Okay, that's going to lead us to our fourth polling question here. Give me a second to post here. And that is going to be do you want additional insight to how tools can help increase communication and collaboration. So, while we are voting on that, folks-- and again, if that's not something of interest or if it's felt well-covered, feel free to vote no. And we can move on. But we want to make sure you get the information you need there.

So, let's get a quick question in while people are voting. Some of this stuff you had covered, but people coming and going. If it's a repeat question, we can go through it really quickly.

Todd Waits: That's great.

Shane McGraw: From Andrew asking what is the major difference between Agile and DevOps. Right now, it seems very similar to the Agile environment that I'm in.

Todd Waits: I agree. I agree. I think for us we see DevOps as extending Agile and formalizing the integrations with other teams and organizations. It's also highly focused on the deployment aspect, the idea of developer's definition of done doesn't just include it works here, it works in testing. It means it's working in production.

Aaron Volkmann: Right, so it's taking Agile and enveloping it so that operations is now Agile, security is now Agile. It's giving agility to a wider broad scope than just getting done with sprints and being able to say we released this.

Todd Waits: Yeah, I think that's a good point. I think focusing on that implementation and extension of it.

Shane McGraw: Again, we had a follow up question asking about the location of the previous webinar. The easiest thing to do at this point is just email info@SEI.cmu.edu, and we can give you that location. Again, a couple questions about where to find the slides for this presentation, if you just look dow9n to the files tab, you'll see all that information there now.

So, let's work in one more question, guys, if you don't mind from Rodney asking what are some common

mistakes organizations make when implementing DevOps, common mistakes.

Aaron Volkmann: Maybe throwing up new siloes of-- we're going to talk more about this a little bit later on, but I think forming DevOps groups that then form new siloes and create new walls and barriers is a big mistake that organizations make.

Todd Waits: I think also the top down approach. You can't make this change by force. It has to be something that everybody buys into. It needs to be supported from the top. But it really happens from the people closest to the work. So, if the people closest to the work are implementing this and they have the support from the people on top, then it will happen. But if either one isn't really supporting it, it will fail.

Aaron Volkmann: Yeah, I think because every organization's unique, and every group within an organization, every little business area, every technology group, each of them are snowflakes. And what works for a web team using the latest and greatest hipster tools to get their jobs done, if we take what they're doing and scale that out across your enterprise, and they reach a mainframe or another legacy system, then that scaling is going to fail because it has to be implemented in a unique way between those two groups depending.

Shane McGraw: Okay so, we have the results from our question about seventy-one percent saying yes, we'd like a little more additional insight, and twenty-nine percent no. So, if we could work that in, as well.

Todd Waits: We can give an example of one other thing. We have a couple of slides prepared for that that we could do.

ChatOps

**024 Okay so, one term that's been coming up over the past few years is this idea of chat ops. And at the-- what you see here is just a sample of a chat window. At the very least, what chat ops is is notifications. The idea is that we're using the chat room or communication tools as a means to

drive, to bring awareness to the development effort. And by awareness, I don't mean to necessarily outside stakeholders, but definitely all the people that are involved in the development itself.

I don't know if you had anything that you wanted to add here. Okay so, this example is actually pulled from something that happened to me this week. It is I did a build for a customer. We have to utilize an air gap solution to deliver one of the pieces of software that we built. And I delivered the build and received this notification.

Aaron Volkmann: You were about to deliver the build.

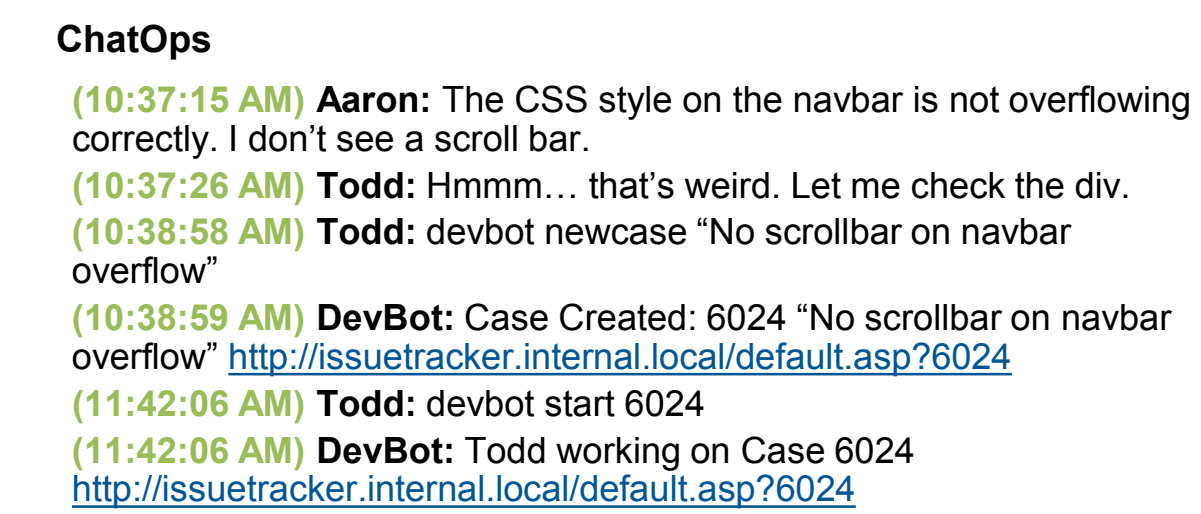
Todd Waits: I was about to deliver the build. And I saw that it said the test configuration. So, I realized that I was actually packaging the wrong build, that I triggered the wrong build. And so, I was able to quickly go in, trigger the production build, and release that within thirty minutes rather than having to-- I have this build. I deliver it. And it goes into a production environment and completely fails because it's pointing at all the wrong servers.

So, this was something that is extremely useful just at a bare minimum is just making sure that our notifications from build servers are hooked up and notifications from issue trackers, that source code and issue trackers can be integrated, things like that. And we can get

those notifications of completion, of test results, and things like that directly in the chat window. It reduces some of the context, which-- we'll get into that in just second.

Aaron Volkmann: I think the magic words here are real-time. It shifts these notifications to real-time instead of an email or checking some place to receive this information.

ChatOps



**025 Todd Waits: So, for example, one of the other scenarios that we had last week was Aaron and I discussing an issue that we had with some of our front end CSS and a scroll bar, or a div not scrolling directly. And so, from the chat window, I'm able to create a new case using our dev bot.

So, this is kind of a more sophisticated way to approach to chat ops is introducing bots into your chat environment. And so, directly from that chat window within the context of our conversation, I'm able to create a new case, and then with that show that I've started to work on that. So, Aaron knows that I'm solving something. It's recorded into our issue tracker which means that other people have transparency and visibility into the work that is getting done and happening.

Aaron Volkmann: I was very skeptical about if this would be worthwhile or why bother writing a bot to be able to issue command line from a chat window. I could just alt-tab over into a web interface to create a ticket. But I've seen that this creates a real-time stream of consciousness amongst the development group, especially when there's many more developers at play so that everybody knows what everybody's doing in real-time. You don't have to go into an issue tracker and run a report to get this information. It's right under your nose in your chat window in a chat room. And I think by letting everybody know what everybody's doing, if somebody's going ahead and addressing something, then it reduces waste because that information gets to all the players in real-time.

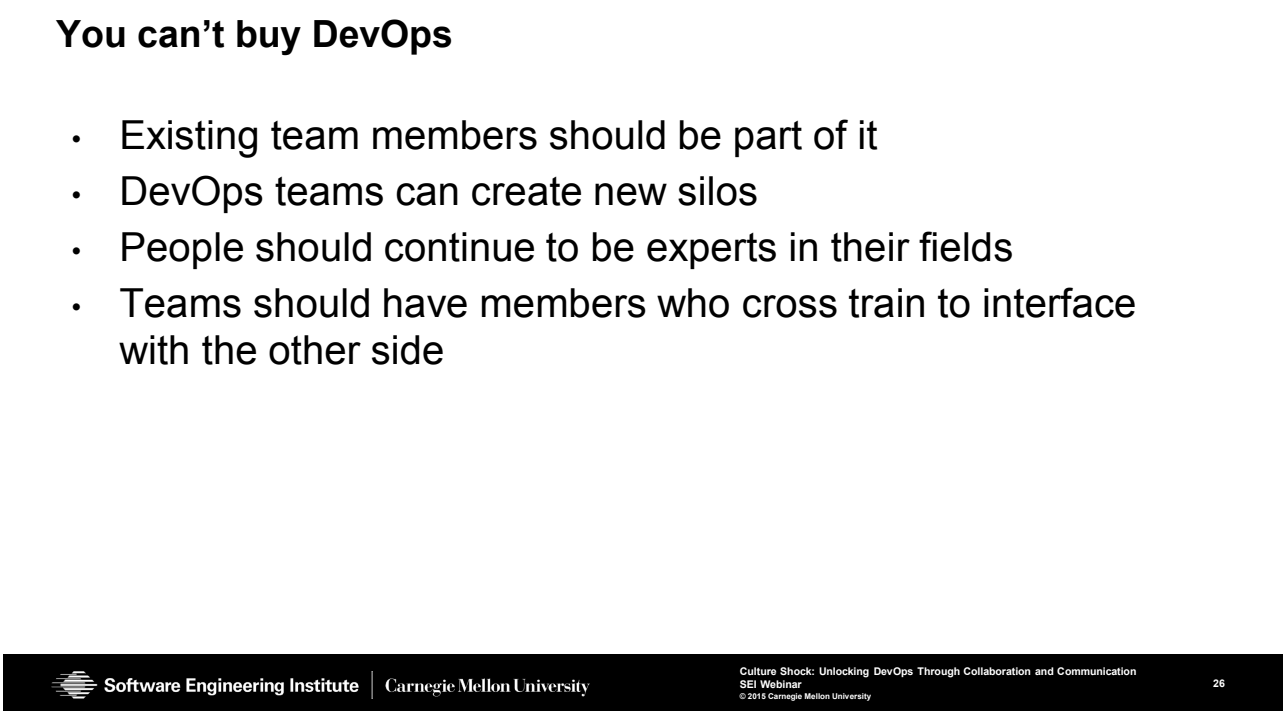
Todd Waits: It's also fun because it becomes a communal command line for the development team with their

tools. A nice thing that it can be used as is it can also be used nicely as an onboarding technique to introduce somebody's skill set, to introduce somebody new and expose them to all of your tooling because they can help write plugins to it and figure out this is how all these systems work together.

Aaron Volkmann: Right, and they're watching everybody else work because they're in the shared virtual space.

Todd Waits: Yeah, it's been very successful for our team.

You can't buy DevOps



**026 Okay so, the last part that we really want to get into is just this idea that you can't buy DevOps.

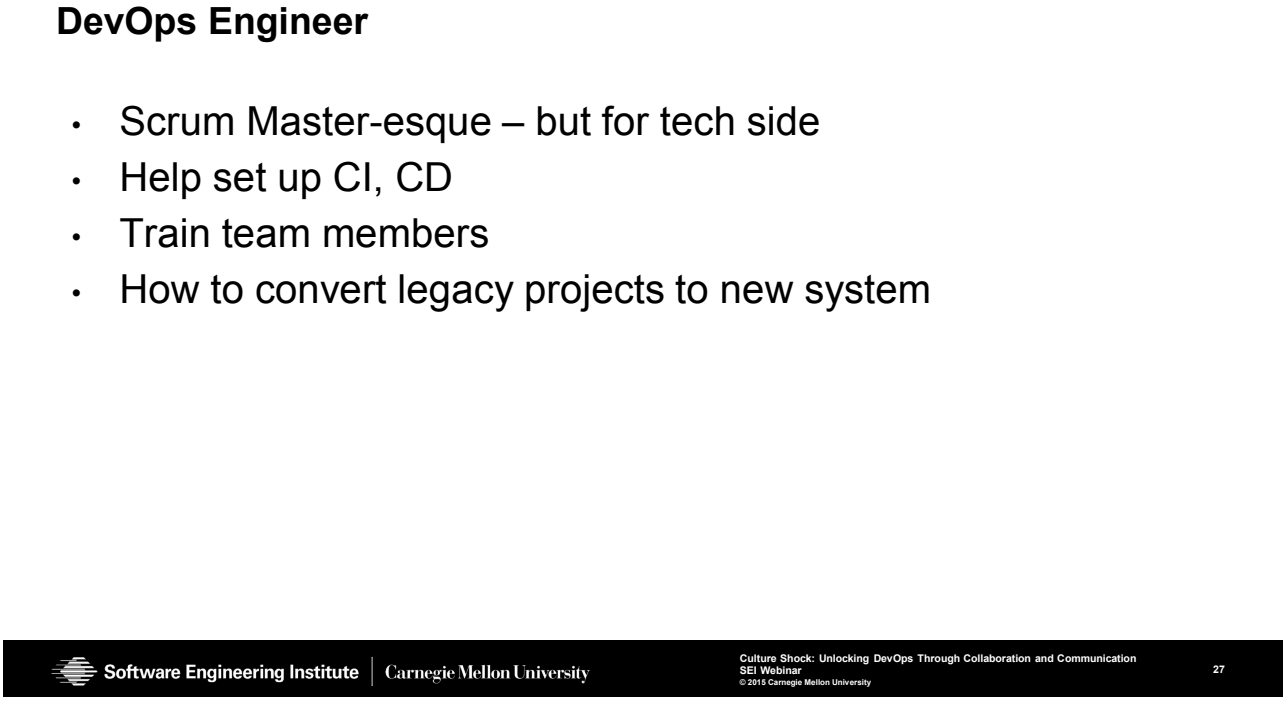
Aaron Volkmann: Right, every organization's unique. Every group's unique within an organization. And you can't just buy some canned solution and try to implement it. That change has to come from within because it's your organization's people who know the most about how your systems work and what it would take to transform areas so things get done in a different new way.

We see people-- developers remaining expert developers and operations engineers remaining experts at what they do. But perhaps within those teams, if we can have intermixed teams with devs and ops within the same team, that's wonderful. If that's not possible, then maybe cross train people to become generalists to serve as ambassadors to reach across the aisle to the other side so that we get that sharing of information.

Todd Waits: Yeah, I really like the idea of being able to-- again, by understanding a little bit of what other people are going through, it creates that shared understanding. It increases empathy in stressful times.

Aaron Volkmann: Right, and also so that each side's aware of what to expect from the other side and be aware of their expectations, their constraints, and be able to work together better.

DevOps Engineer



**027 Todd Waits: A lot of what we see now is we see on a lot of the job boards is this DevOps engineer.

Aaron Volkmann: Right, right. If you go over to Stack Overflow careers and search for DevOps, you'll see tons and tons of job openings for DevOps engineer.

Todd Waits: It's kind of a contested position because it's difficult to say whether the DevOps engineer is a developer with sys admin background or if they're a sys admin with a little bit of understanding of coding and development practices. The way that we started to look at it is more of role that it becomes like a scrum master, where a scrum master is a role within Agile scrum implementation. That ensures that the methodology is

followed precisely. A DevOps engineer could be a role that ensures that DevOps practices are followed properly from a technical standpoint, somebody that can help and train groups and bring awareness.

Aaron Volkmann: Right, I see them bringing continuous integration, continuous deployment, setting up those systems in an organization, assisting with maybe converting existing projects, re-architecting existing projects so that they can take advantage of automation technologies as well as maybe set standards for future development so that they can take-- so that those projects can take advantage of new automation tool sets that you might be using. We see this person as an agent of change within the organization to change what's already there, not really to create their own silo and be a service provider.

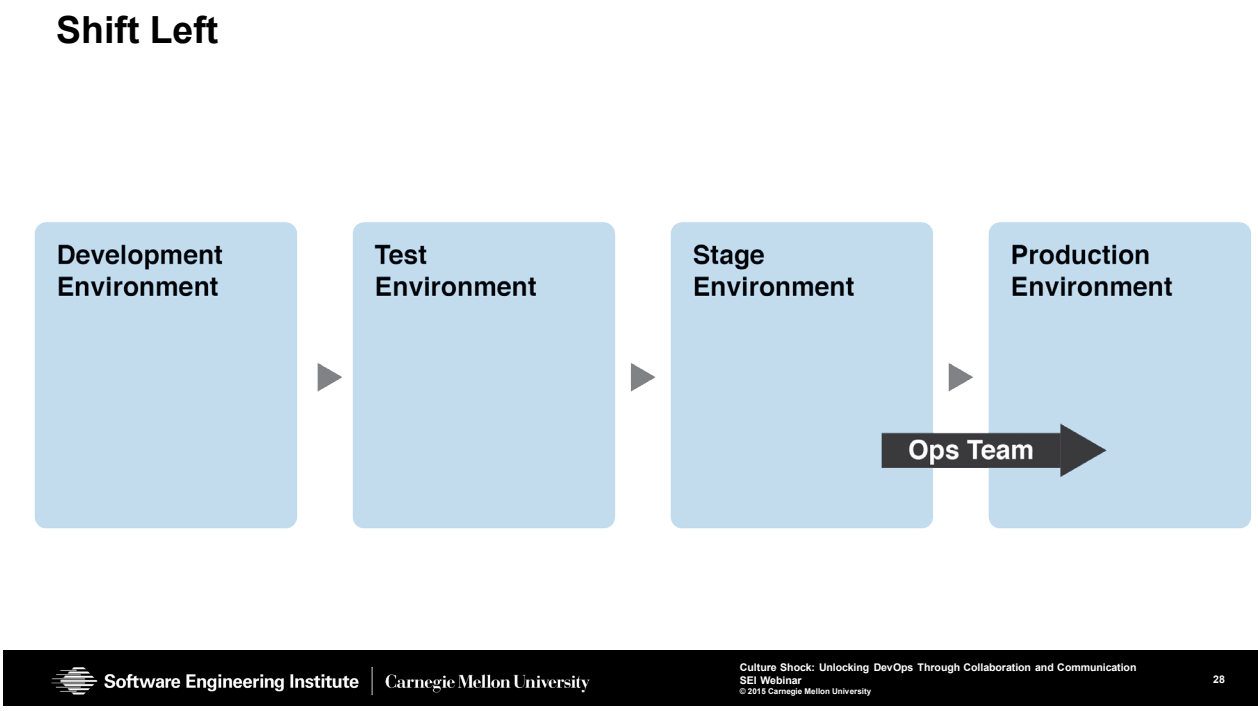
Todd Waits: Yeah, it's important, again, that we're integrating everybody, that everybody's a part of the same team with aligned business goals and objectives. So, to bring in a DevOps engineer and just say no, your job is still to develop, your job is still to make sure that operationally our application's maintained, and your job is to implement--

Aaron Volkmann: Yeah, submit a ticket to DevOps engineer to get your project into continuous integration.

Todd Waits: Yeah. That ends up causing a lot more pain than it's worth.

Aaron Volkmann: Right, we're trying to create a capability within an organization.

Shift Left

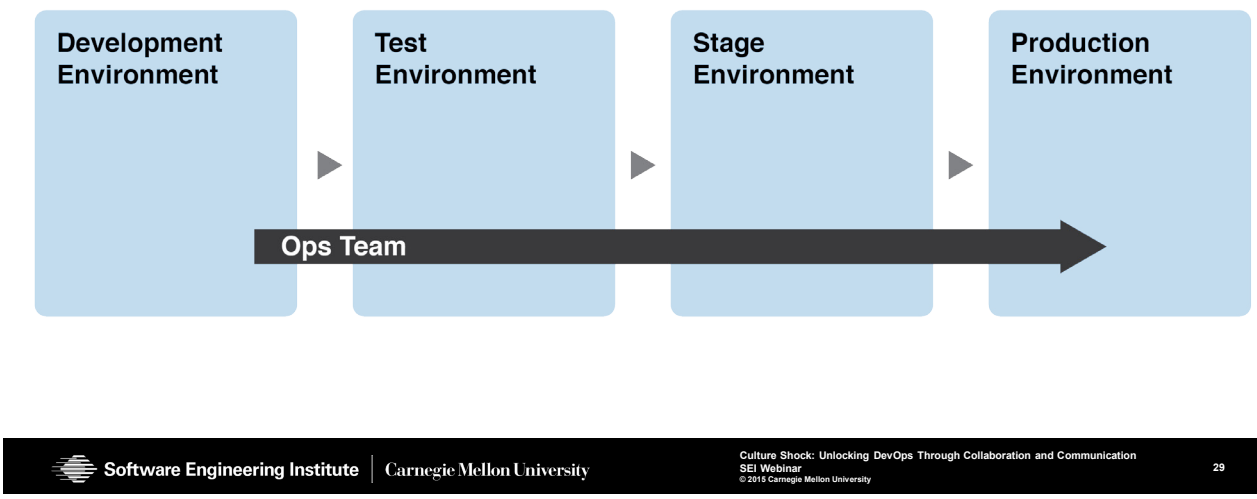


**028 So, it's really hard to talk about DevOps without talking about shifting left. We say it all the time. So, what we mean by shifting left, here you see we have project timelines. It goes from dev all the way through to prod. And typically, before we started thinking about DevOps, in many organizations the ops teams concerns only came into play once we're ready to step into the production environment. This goes for different types of testing,

operational acceptance testing, testing the fail over, maybe even looking at security stuff right there at the end.

But through shifting left, we're going to be--

Shift Left



**029 Including these concerns about the operations team, perhaps security, earlier in the process to lead to better outcomes. I think this has to do with testing, stuff like web ports are open, for a really simple example. How much disk space will we need? What version of the operating system does this have to rely on?

I think this becomes even more-- this is a big deal for the government to prevent to healthcare.gov type

situation where we have many contractors developing units together, then only whenever we're ready to go to production did we integrate everything together to test and lead to a very costly failure. But if we could shift left those concerns, then we could take care of those concerns up front and avoid those types of mishaps.

Todd Waits: Yeah, by including those integration testing and operational testing at the beginning, we can be building toward success.

Aaron Volkmann: Right, and just to underline again, systems should be built with security in mind from the start. So, I know we're talking about development and operations, but this also includes security. Security should be at the forefront of thinking about a project from its beginning.

Todd Waits: The DevOps really came out of the idea of bringing development and operations together. But that's just part of the story. It's really any organization or group that may have an impact on that final product, make sure that they're brought in early in the process.

A couple of anecdotes from my experience here that this shift left process has really come into play was one was for our expert control things that we have to do. We have to make sure that we go through legal process whenever we're releasing a tool to one of our sponsors. And initially, we would be releasing that

tool and suddenly, wait no you cannot release this tool until it goes through legal and gets the proper markings on it so that it can be released properly. And so, as we've got experience with that, we're able to involve our export control group earlier on in the process to make sure that we're taking care of that so by the time we hit our deadline, we don't suddenly have a delay of two months while it goes through review and control policies.

Similarly, one of the sponsors we've-- well, let me share a different example. In the past, I've worked with other organizations where we were building a system for them. And when we go into integrate it, suddenly their operations groups aren't prepared for us to put that in. and so, we've been collaborating with the stakeholders in that organization, but not necessarily involving all the key stakeholders from that organization that are impacted by us deploying. And that was an oversight on our part. Once we made the switch to make sure whenever we instigated or started a relationship with a customer in that company, we were able to involve the IT and operations group from the beginning. And that way, they didn't feel as if they're toes were being stepped on or that we were overreaching our bounds or trying to dictate how they should do their job. They had the proper input that they need to have.

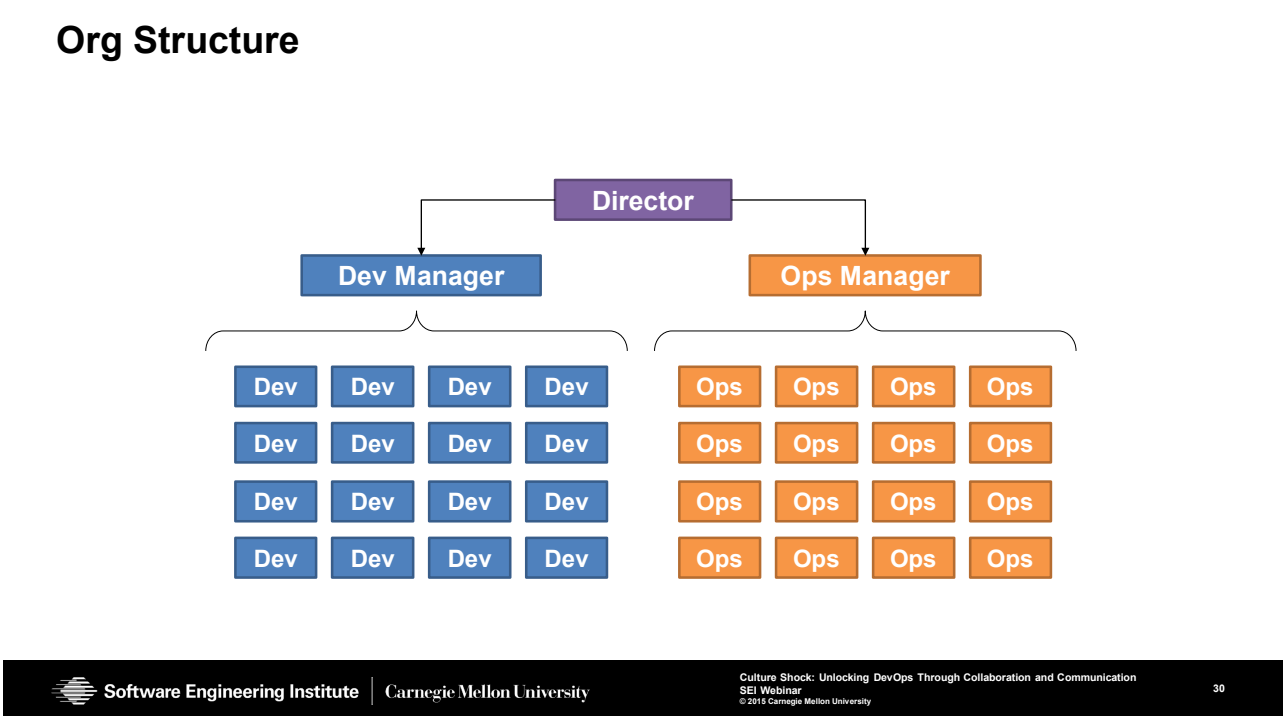
And rightly, so they were upset that we were just coming in and saying

here's this thing. Deploy it. They were able to have that and ease that process of transition when working with outside, as a contractor, working with outside groups.

Aaron Volkmann: Right, so by shifting left, we'd have-- it would buy us months to let these sort of relationships work themselves out instead of at the nineteenth hour whenever we're ready to meet our deadline.

Todd Waits: Right. And then suddenly we deploy and nobody's ready to actually receive it. That was a big success of implementing a DevOps approach to this.

Org Structure



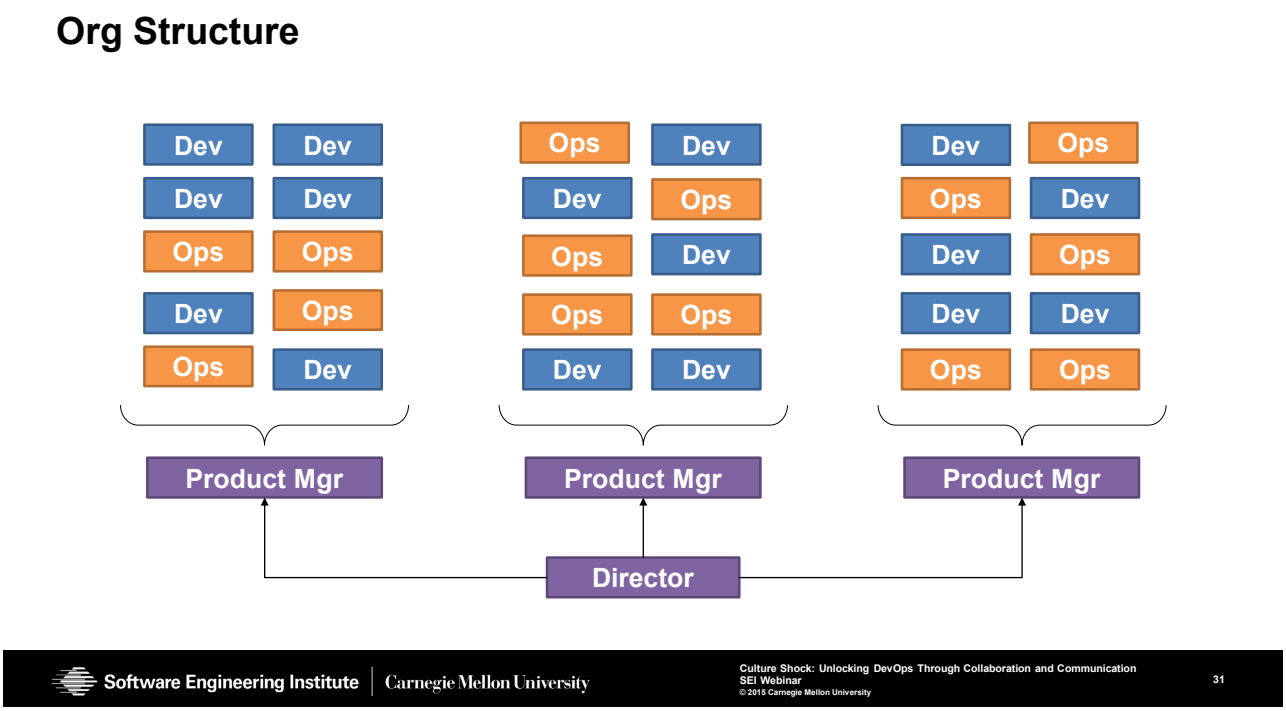
**030 So, going to-- we talked a little about organizational structure

earlier. We want to go into that in a little bit more detail. So, here we have, again, that top down approach. Traditionally, we have organizations that have been split along functional teams. So, we have a director with a developer manager and an operations manager. The developer manager, his goal, his primary goal and object is to, again, introduce new features that customers are going to be wanting to use working with business analysts to make sure that we're getting the right things in there and introducing change. And the operations manager almost has the exact opposite goal where his goal is to make sure stability is the primary factor in any development project. And so, development naturally introduces instability, and operations naturally reduces instability. And so, that is looking at how organizing our teams along functional lines can create these divides.

Aaron Volkmann: I think misaligned-- a big danger here I think is misaligned priorities. There is a data warehouse manager that I'm friends with, I worked with in the past who worked on the operations side. And he said he felt like a short order cook. Every time his phone rang, there was another request from the business asking for something new. So, his team, his operations folks, would be pulled in to fix the squeakiest wheel. So, even if they were embedded in a development team, they would be pulled off of that development project in order to work on break fix, put out fires,

satisfy immediate demands. So, that's one of the big pitfalls that we see with an org structure that looks like this because we have the danger of that sort of thing happening.

Org Structure



**031 Todd Waits: So, one of the things we like to see is when we can organize groups from a bottom up direction where we have our teams organized around products, around initiatives. And the leadership on the bottom is there to support the work that's being done. Everybody's there to support the delivery of business value as opposed to my job is to deliver this new features. My job is to deliver value to the customer.

Aaron Volkmann: Yeah, in this kind of org structure, everybody in one of

those columns is all rowing the boat toward the same goal. They all have the same leader who is making sure that that happens and is able to balance out conflicting priorities to avoid conflicts from happening to assure that everything happens on time when it needs to.

Todd Waits: And a lot of times, I know that we're putting this up here. So, I can anticipate that some people are saying it is not realistic for us to implement a brand new organizational structure in our company, in our group. We understand that. And so, going back to what Aaron was talking about in the previous slide is that the important thing is can we align our incentive and our priorities along these types of lanes so that they are aligned by initiatives and projects rather than by the function of the person.

Aaron Volkmann: Perhaps in the future, Gene Kim is just to quote him. He was saying, "In the future, we're always have centralized operations teams. And if they don't like this on the screen, and if they are all together under one manager, then instead of responding to request tickets to do little chunks of work, what they should be spending their time doing is building automated capabilities to provide their users with self-serve capabilities. So, if you need a virtual machine, build a capability so that the devs can ask for it themselves or the business can request it themselves. That's where

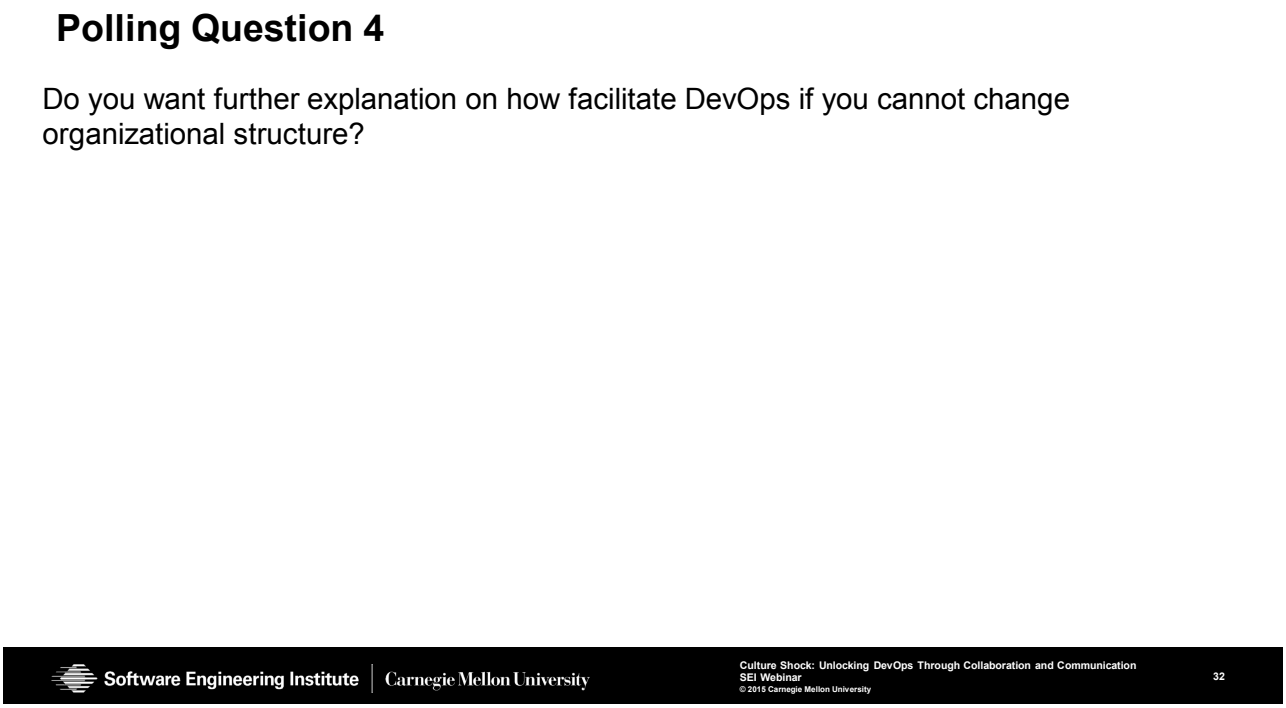
we see the future of centralized IT going. They're still there, but shifting the type of work they do from doing things manually to building capabilities that provide economies of scale so that doing these types of things that happen manually in the past can happen automatically with very little human labor.

Todd Waits: Right, and there's a lot of tools out there that can do a lot of that, speaking of provisioning virtual machines on demand. By just setting a few parameters and hitting go, you can have a working virtual machine without any human intervention already. And so, that type of thing can be extremely helpful to development processes and can also provide a standardization because that way, you don't have people that are doing one off virtual machines and perhaps introducing security vulnerabilities into the work flow.

Aaron Volkmann: Right inconsistencies, that sort of thing.

Todd Waits: Exactly.

Polling Question 4



**032 Shane McGraw: Okay so, this is going to get us to our fifth and final polling question of the day, folks. We'll pose that now. And we'd like to know do you want further explanation on how to facilitate DevOps if you cannot change organizational structure.

So, while we're voting on that, let's dive into back into a question. A little bit of a longer here from Steve, but wanting to know developers supporting operations invariably encounter operators' unanticipated issue-- or uses of a delivered system. How can these be communicated with the program without being seen as defects, deficiencies/blame in either the development or the requirements processes, especially by those middle managers so very, very

heavily invested in those past processes?

Todd Waits: That's a very good question.

Shane McGraw: Do you need a repeat, or were you able to gather all that?

Todd Waits: I think I'm good.

Shane McGraw: Go for it.

Todd Waits: So, I think in some ways we have-- I'm going to repeat it and hopefully I get the gist of it. So, the idea that we have things that are being reported as bugs or deficiencies but that they may just be people using the application in an unexpected way, which isn't necessarily bad. It's not necessarily good. It's just unexpected. And so, how do we communicate there's this unexpected use going on in a way that doesn't say you fouled up, you've messed it all up.

Shane McGraw: That sounds correct.

Todd Waits: Okay so, I think that goes back again to the unified goals and objectives. If we have an integrated team that has their priorities aligned, then we can communicate in a way that doesn't-- that's not wielding a hammer. It's just notifying we're seeing this type of activity. Our goal is that we're delivering business value. So, if we have a piece of our application that's

not meeting the business value, then our job's not done yet. And so, it's not that you've done it wrong or you've made a mistake. It's couching in a form of we have to continue. We're just not finished yet. And so, let's keep iterating at it. Let's add this to the next iteration and do better this next time and we can kind of patch these holes and fix this.

Aaron Volkmann: I think the idea of up to date documentation would also aid this where information is shared and available through automated systems.

Todd Waits: Yeah, I think that's a good point.

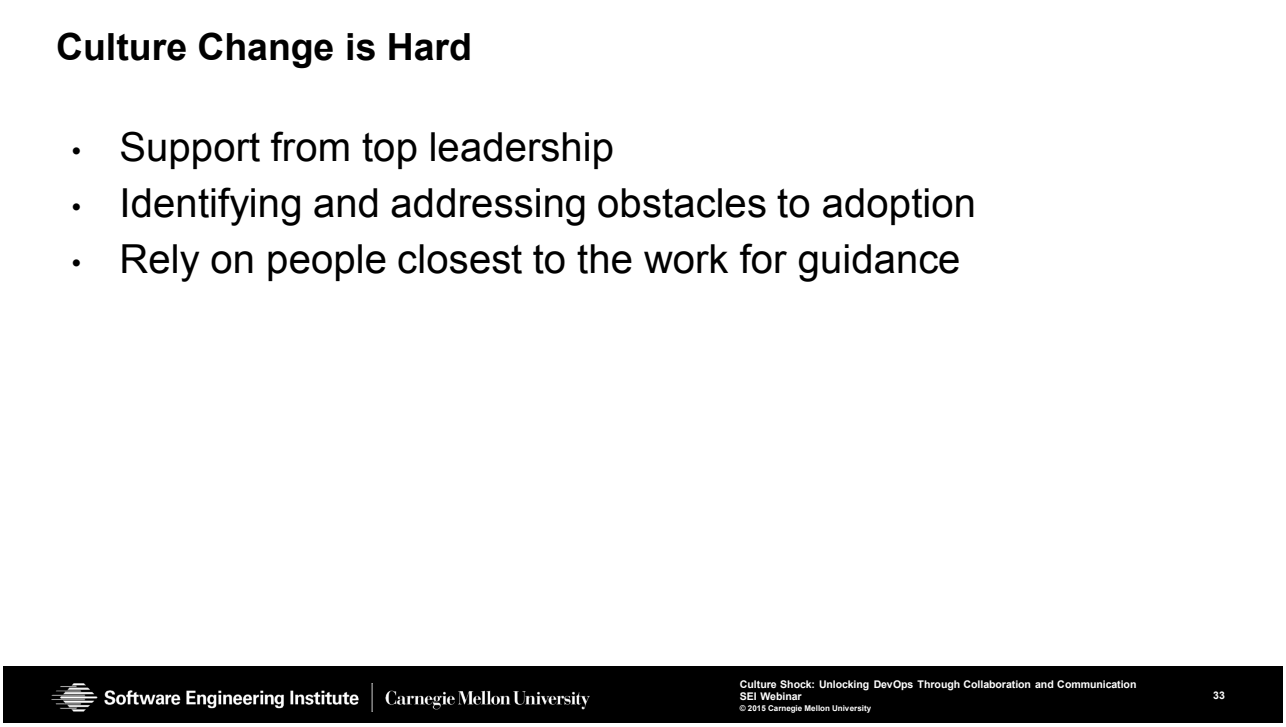
Shane McGraw: Okay so, let's share our results. I think we know what the answer's going to be.

Aaron Volkmann: Yeah.

Shane McGraw: We have a sixty-nine percent yes and thirty-one percent no. so, can we dive into some further explanation there?

Todd Waits: Absolutely, I think for us, understanding that culture change is hard--

Culture Change is Hard



**033 That it's-- regardless of whether you can easily shift your organization around or not, culture change is going to be a difficult thing to make happen. And so, one of the things we can do is to look at how can we justify the use of some of these methodologies.

Aaron Volkmann: Right, it's all about convincing top leadership, C-level leadership to come along with this. One way that I've seen organizations do this is by creating sort of a skunkworks team, carve off a little project with a team that's organized in this new way using new processes, tools, and technologies and gathering data to prove that this can-- data that shows that there is a benefit to doing things with a DevOps philosophy in mind.

Todd Waits: I think as far as the data goes, it's important to remember that if you're not tracking anything, you should start. It doesn't really matter at this point what you start tracking, but just start tracking something, whether it's number of cases resolved, whether it's time to go to production from initial build. Any kinds of these metrics that we can show we went from resolving fifteen cases a week to resolving forty cases as week, or we reduced time to deployment by three months.

Aaron Volkmann: Right, I think go for the low hanging fruit because any data that's real is good data. And I wouldn't stress about trying to find the perfect metric to demonstrate success because success can be visible in many unexpected places.

Todd Waits: And once you have the metrics, it's difficult to-- it's difficult to say I don't like this process. Well, you may not like the process, but the results are there. And so, whether we like doing a culture change, whether we like implementing new things in the organization, we see the return on that investment very clearly. And the more that we can document that and show that, then the more success we'll have gaining new followers, new adherents to the new methodology.

Aaron Volkmann: Another aspect is there's lots of published research out there showing that these things do work. Gardiner said-- these things work. They're mainstream. They're

here to stay. Gardener Research said by 2016, twenty percent-- twenty-five percent of the global two thousand organizations will have implemented DevOps within their organization. Computer Associates commissioned a study that says-- the participants said they had a nineteen percent increase in revenue, a nineteen percent increase in quality and performance, and twenty-one percent reduction in time doing rework and fixing bugs whenever they implemented DevOps philosophies. Puppet Labs 2014 state of DevOps poll showed that companies reported they doubled their successful change rate. They increased their lead times by eight thousand percent. And they said they felt that they were two times as likely to exceed in profit, market share, and productivity goals. So, if you were a C-level leader, could you think of implementing DevOps philosophies in your organization, if there's research out there saying it can do this, should there be a higher priority for you?

Todd Waits: I think also if we look at organizational barriers, a lot of times one of the things that we started doing was providing kind of tech exchange type lunch where we were able to present new technologies or just interesting things that we saw in the development field. It allowed us to start getting a handle on what was happening in community while-- so maintaining our skills while interacting with each other in a very informal setting where we were able to show and build trust

with one another that we're all out there building skills. We're all out there finding what's really going to help improve our state of practice and the state of practice in the community at large.

Aaron Volkmann: So, that's what we should do until we purchase the whole Rock Band setup to build trust?

Todd Waits: Right, so if you can't bring Rock Band to your organization, at least you can start sharing knowledge back and forth and creating these informal, low-stress environments that can build trust amongst everybody. I found that that's helped a tremendous amount for us building that trust with one another.

Shane McGraw: Okay so, that's it.

Q&A

**034 We're on to our Q and A. So, before we dive into our final Q and A with Aaron and Todd, just a couple housekeeping items. A couple of questions have come in asking where a replay can be found or where they can watch the archive. So, people are ready for an encore already. So, you guys must have did something right. The file, most likely the archive will be up at some point tomorrow. And you just use the same login credentials that you used to register today. So, you just go back to the same URL and login. And the archive, like I said, will be up some point tomorrow, most likely, if not, definitely by Monday. And you'll get an email with that location.

A couple people, again, asking about slides. They're in the file tab. You can

walk away with that along with other work on DevOps from the SEI workshops, training, processes, things of that nature.

Again, one more plea, too, like I said, a lot of questions that we'll get into here in a second. We have about five minutes left. If we don't get to everything, we invite everyone to join the LinkedIn group, the DevOps forum on LinkedIn. If you have a LinkedIn profile, just go to LinkedIn. Search under groups for DevOps forum, SEI DevOps forum. Post your questions there. We look to keep this conversation going with you guys there.

So, let's get into the questions now from Damian asking what should be the role of enterprise social networks in the DevOps equation?

Todd Waits: That's a good question. I think in the same way that we're bringing in transparency and communication collaboration. I think if you're finding that a tool's not being used, it's really important to identify why it's not being used. Does it not-- does it just not fit into the workflow? What is the actual breakdown that may be happening?

I think enterprise social networks can sometimes fit in that. I think either they become successful, and everybody'd on their Yammer profile talking, or they just kind of languish and wither.

Aaron Volkmann: I think they're useful to help bring your organization together, especially if you're in a very large organization, to try to put a name with a face, and to be famil-- depending on what's on your social network profile, but knowing what the background of everybody and hoping-- I think it's a transparency thing to try and increase the culture of collaboration within your organization to know something about each other that's searchable and accessible.

Todd Waits: Yeah, and just a place to be able to share those thoughts and define the common grounds.

Shane McGraw: Okay, before we get to John's question, just a reminder to everyone as well to fill out the survey upon exiting, as your feedback is always greatly appreciated. So, John wants to know what's your experience in creating goal and performance alignment across dev, test, and ops, assuming that they are three separate groups. What activities do you perform to ensure the tweaking of the performance management system to reinforce the new behaviors?

Aaron Volkmann: I think holding everybody accountable towards deadline dates and things like the number of defects discovered.

Todd Waits: I think trying to also look for-- as you're going through iterations is looking for the blame game. If you find things that are

being punted back and forth with this isn't acceptable and all you see is the back and forth, we can identify those areas as we can improve this.

We can get people seated down talking to each other and communicating so that we can understand that when-- testers, when you're handing this over and you're handing back to ops and it's-- and operations, when you're working with developers that you're working together to solve these problems. If all we're seeing is these, again, maintaining siloes by passing things ba-- or artifacts back and forth to just be fixed, the not my job. In DevOps, it is your job. Everybody's job is to make sure that there's functional business value delivered to the customer.

Aaron Volkmann: And I think also making sure everybody's assigned to be a member of that team in the performance management. I know a lot of organizations where it's the developers who own that project. And the operations team is merely there to act as a support role or to provide a service. So, getting operations as one hundred-- full-fledged team members on a project and rating their performance that way would be a good strategy to rate them that way.

Todd Waits: I think so because otherwise you run into again the not my job. If I'm not being judged on my performance in this team, then it's not my job to perform in this team.

Aaron Volkmann: Right, there's no incentive.

Todd Waits: Use it-- if you want tools to be used in the organization, then it sounds kind of hokey to say this, but they have to be used. You have to show the outputs of those tools in order to have people utilize them. So, if people aren't seeing the results of the reviews used or the results of issue tracking highlighted, then they're not necessarily going to use those.

Aaron Volkmann: Right, nobody's going to take their time out to enter some data into a seemingly a black hole that goes nowhere that's never talked about again.

Shane McGraw: Okay, we've got about a minute left. We're going to try to squeeze in two questions here. From Wally asking how do the actual roles within the development areas and operations change when pursuing DevOps?

Aaron Volkmann: I sort of think the roles stay the same. The developers remain developers. Operations remain operations. Perhaps, certain team members on a development team know a little bit more about what it takes to run something in production. And operations might pick up some scripting and development practices just so that they can help automate.

Todd Waits: I think hopefully, you're just talking to different people,

the new people, that it's not a whole overhaul on your role in the organization. So, me suddenly as a developer I have to learn everything there is to know about Linux, but it's that I'm now interacting with people. And I have that input in my job, and they have my input in theirs.

Shane McGraw: Okay, and I think this is more of a tool question if we can get a quick answer though from Regis asking how do you enable close collaboration among globally distributed dev and ops teams across multiple sites.

Todd Waits: For me, I think it goes back to transparency and using your communication tools. Everybody should be in a chat room together. You can have individual one off chats if you-- because you don't want to have everybody listening to you talk about your puppy. But if you have on person who's interested in your puppy, well then go and chat with that person. But everybody in a team should be in a room together interacting, talking about the product, discussing problems. You'll find surprising sources of solutions that you did not think would come if you're all working together in the same place.

And do you have anything to add? I know we don't have much time.

Aaron Volkmann: Yeah. I guess I'd just like to add that nothing beats face to face or voice, real-time

interactions in that you can't replace that with emails.

Todd Waits: I think if you need help, share a desktop. There's plenty of technologies to share your desktop across continents. And so, make use of those technologies to talk face to face, to show your desktop, show what you're working on, show where you're seeing the problem, and interact with people that way, not just with an email and a screenshot, really interact with individuals.

Aaron Volkmann: Yeah, the big problem is with the time zones. And I don't think our research has progressed far enough to be able to help out with that. But check our LinkedIn group and for future--

Todd Waits: We'll further these discussions there.

Shane McGraw: Aaron, Todd, great presentation. Thank you. Folks, that's all the time we have for today. We want to invite you to the next webinar. It will be May 7th. And that will be using Did Fail to analyze flow of sensitive information in sets of Android apps. So, again, thanks for joining us today. Have a great afternoon. Please be sure to fill out our survey.