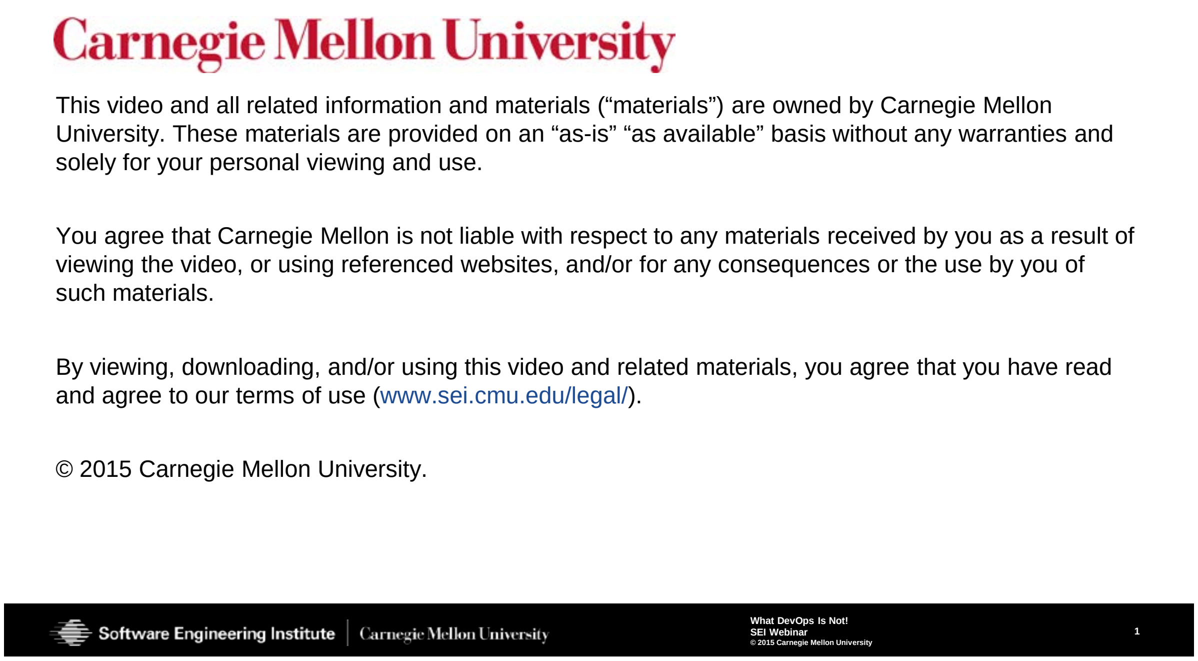
What DevOps is Not!

Table of Contents

Carnegie Mellon University Notice



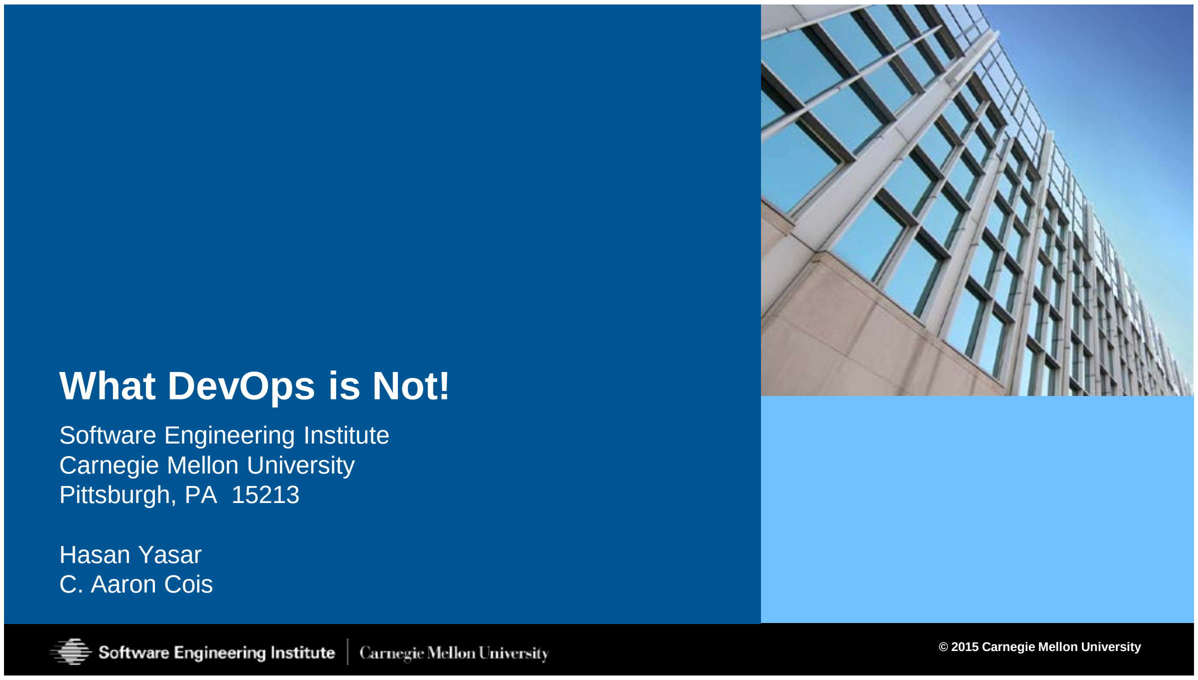
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

1

**001 Shane McGraw:

What DevOps is Not!



What DevOps is Not!

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Hasan Yasar
C. Aaron Cois

Software Engineering Institute | Carnegie Mellon University

© 2015 Carnegie Mellon University

**003 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 What DevOps is not. We'd like to thank you for attending.

Depending on your location, we want to wish you a good morning, good afternoon, or good evening. My name is Shane McGraw. I will be your moderator for today. And again, 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. And they will appear as a pop-up window on your screen. Our first polling question is how did you hear about today's webinar.

Another three tabs I'd like to point out are the files, Twitter, and survey tabs. The files tab has a PDF copy of today's presentation slides along with other DevOps related work from the SEI. The survey tab we request that you fill out at the end of the presentation as your feedback is always greatly appreciated. For those of you using Twitter, be sure to follow @SEInews and use the hashtag SEIdevops. Once again, be sure to follow @SEInews. And the hashtag for today is SEIdevops.

Now, I'd like to introduce our presenters for today. Hasan Yasar is the technical manager of the secure lifecycle solutions groups in the CERT division of the SEI. Hasan leads and engineering group on software development processes and methodologies specifically on DevOps in development. He researches advanced image analysis, cloud technologies, and big data problems while providing expertise and guidance to SEI clients. Hasan has more than twenty-five years' experience as a senior software engineer, software architect, and manager in all phases of software development and information modeling processes.

Constantine Aaron Cois has been developing software for over seventeen years. He received his MS and PhD from the University of Pittsburgh for work developing novel systems and algorithms for medical image analysis. His doctoral research involved developing frameworks, algorithms, and data structures to officially perform complex statistical calculations on large amounts of data. His background in machine learning and complex systems combined with his passion for web-based technologies led him to work on developing complex software systems for numerous organizations including UPMC and Carnegie Mellon University's Software Engineering Institute.

Now, I'd like to turn it over to Aaron Cois. Welcome Aaron, all yours.

C. Aaron Cois: Thank you, Shane.
So, today we wanted to talk about a topic that a lot of you have probably heard a lot about in the last few years. That's DevOps. We've heard a number of different definitions of DevOps. Some of them conflicting in ways. So, we thought we'd take a different approach to it today and talk about what DevOps is not, some common misconceptions we hear about the concept of implementing DevOps within organizations or teams.

DevOps is not

DevOps is not

A FAD

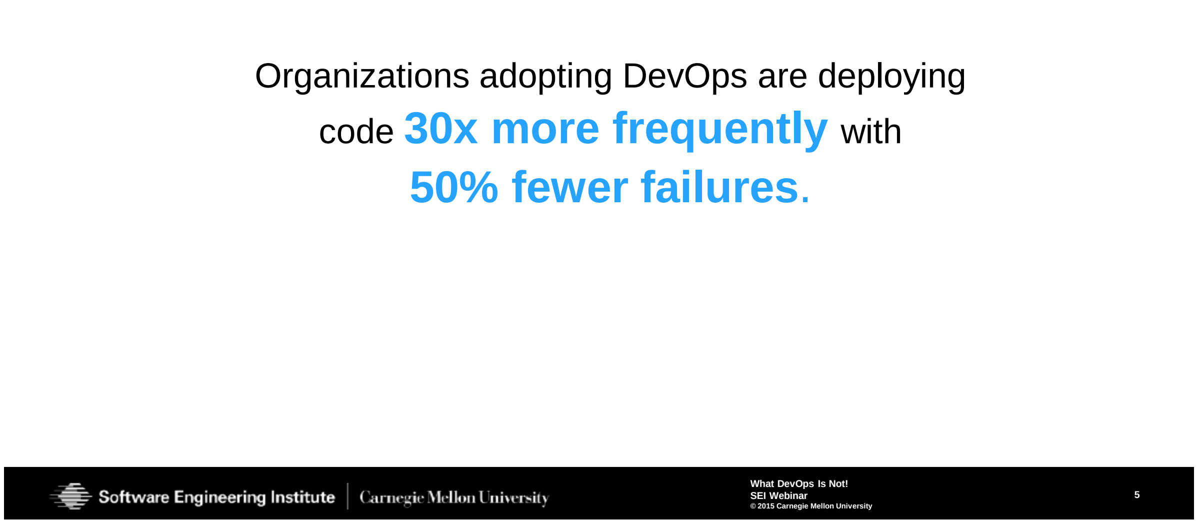
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

4

**004 So, the first thing-- the first most common misconception that I hear is that DevOps is a fad. DevOps is actually something--

Organizations adopting DevOps are deploying code 30x more frequently with 50% fewer failures .



Organizations adopting DevOps are deploying code **30x more frequently** with **50% fewer failures**.

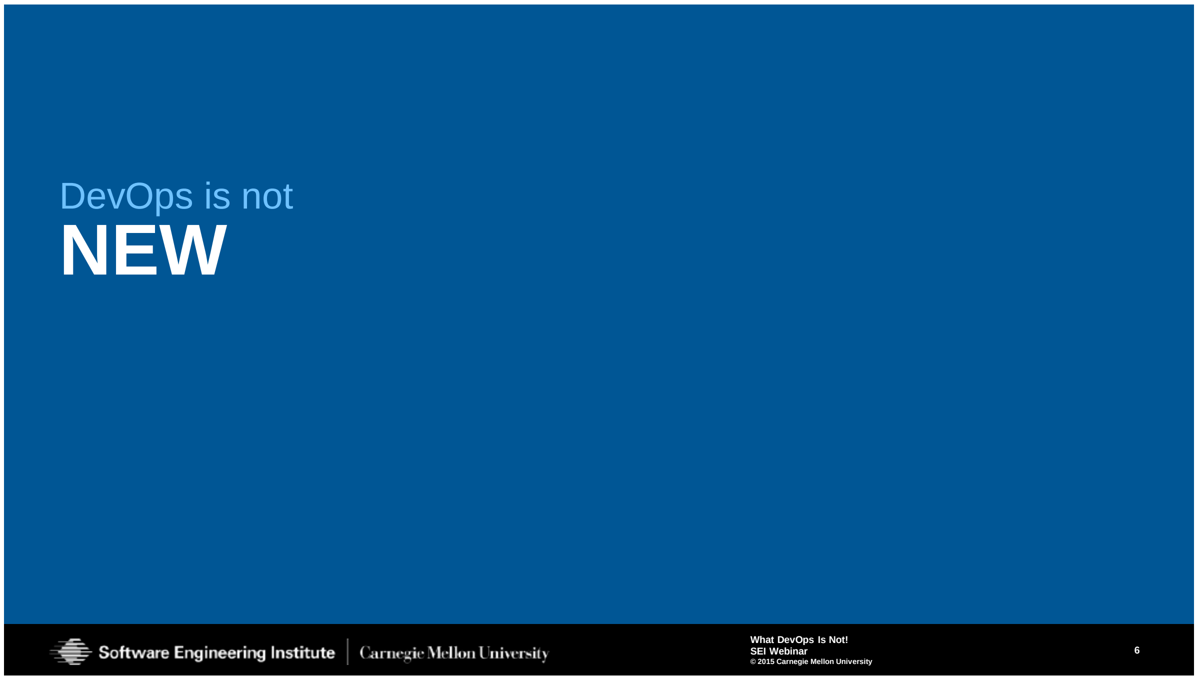
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

5

**005 That's been around for a while and is really providing a lot of value to organizations. A lot of people seem to think that maybe DevOps is something that can work for organizations out in the Valley doing continuous deployment, organizations that are operating in a very fast and Agile manner, maybe not enterprise ready, maybe not ready for business in that way. But a survey in 2014 from Puppet Labs and ThoughtWorks showed that organizations that are adopting DevOps now are deploying thirty times more frequently with fifty percent fewer failures. Those are numbers are hard to ignore. And those numbers mean that DevOps is going to be here to stay. And we're going to see more of it in years to come.

DevOps is not

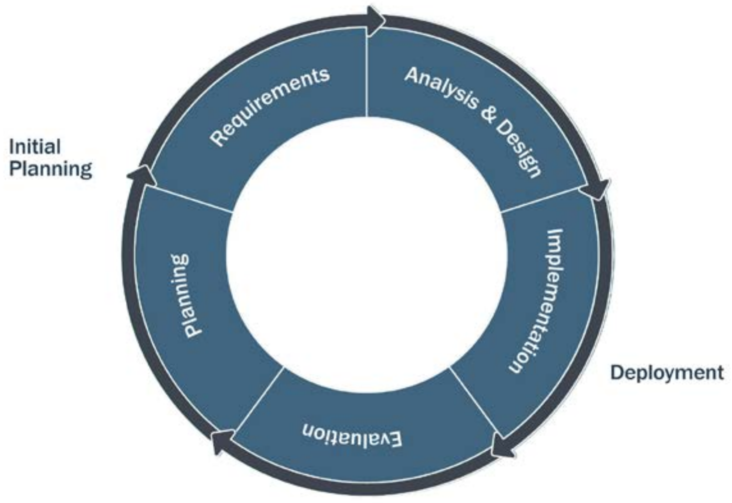

DevOps is not
NEW

Software Engineering Institute | Carnegie Mellon University
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University
6

**006 The second misconception we wanted to talk about today is that DevOps is a new thing, a concept that is absolutely new to the field and is maybe not tried and true, tried and tested.

DevOps is Agile

DevOps is Agile



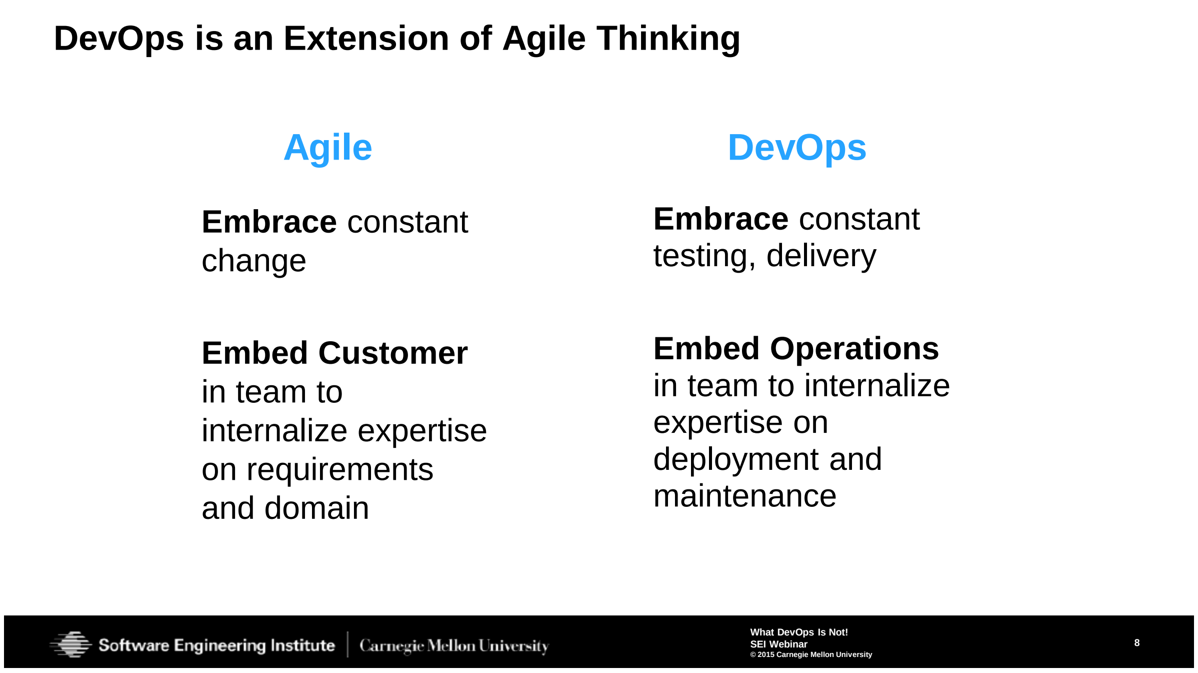
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

7

**007 That's fundamentally not true as well. DevOps is actually highly related to Agile development processes. So, as we can see, Agile introduced the concept that we need to iterate in software development, that we need to move through our cycles in an iterative fashion, that we need to have incremental changes and adapt and react to change quickly and flexibly to make sure that we hit our targets correctly. Basically, Agile said that just doing our planning and then completely planning, moving on to development, completing development, moving on to testing, that waterfall model, wasn't going to work because we were going to learn more things as went on through the software development lifecycle. And those things we learned needed to be adapted to an embedded into our development process to make sure that we changed to meet the changing requirements of our software system.

DevOps is an Extension of Agile Thinking



DevOps is an Extension of Agile Thinking

Agile	DevOps
Embrace constant change	**Embrace** constant testing, delivery
Embed Customer in team to internalize expertise on requirements and domain	**Embed Operations** in team to internalize expertise on deployment and maintenance

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

8

**008 So, if you think of that kind of thinking, think of what the Agile methodology brought to us and brought to software development, Agile's concept was to embrace constant change. It was to embrace the fact that the requirements will change, that the development team and the customers will learn more about their requirements for the software system as they move forward.

So, in order to do that, methodologies like Scrum employed something called a product owner. The concept was to take the expert in customer needs, the expert in requirements, and embed them in the development team as, say, a product owner. To be there to provide real-time instant feedback on those concepts. If developers had to make a decision that may impact requirements, or if they tried out prototyping a specific feature, they could immediately turn to the customer expert and say, "Does this meet your needs? Does this make sense with what the customer's needs are, what the business needs are, etc.?" And the customer could immediately answer.

Then the developer could have the real-time feedback and move in the right direction without having to wait for an answer, without having to do unnecessary work and then undo it when that feedback came in later.
So, this increased the flexibility of the development lifecycle and also increased the pace, allowing more features to be developed more quickly because that feedback was continuous.

Well, when DevOps came on the scene, the primary idea was remarkably similar, to embrace constant delivery, to embrace testing, to embrace operations. The idea was that info about operations, the expertise about the needs of the production environment, the needs of the customers with regards to actually serving an application in production and delivering features to users, that expertise was within the staff that maintained the production systems. That was in the operations team.

So, the DevOps answer to that was to embed operations into the development process. Take experts from the operations teams and put them into the product team. Make them involved in every decision so that at any point in time, developers could turn to them and ask will this impact production, will this impact deployment, will this impact delivery. And again, real-time feedback could be injected into the development process to make sure that development didn't go awry, that it was always consistent and tracked towards operational realities and operational needs. So, really, DevOps is just an extension of Agile thinking applied to operations and sustainment, and also, as we'll see, applied to business needs and business requirements on the far end.

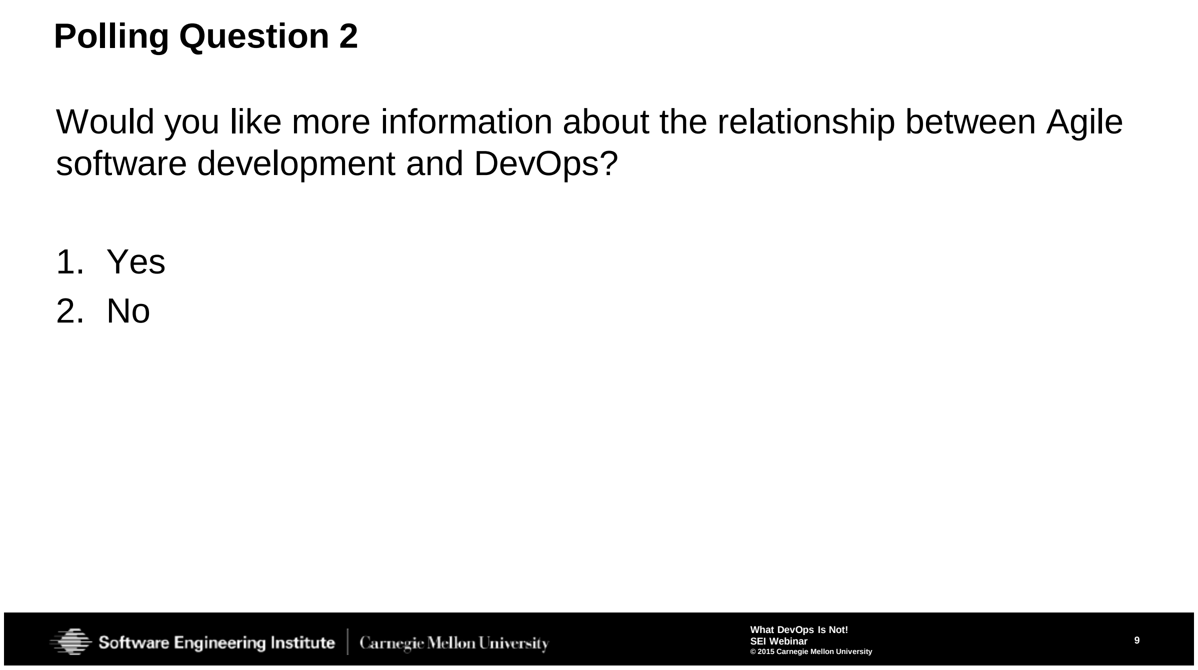
Hasan, do you have anything to add to that?

Hasan Yasar: Yeah, actually, I had a couple things to say about the business requirements. So, if the business people, when they're involved in the development process, they would like to get some type of like hands on, and then get ideas, and also direct the developers and see where it goes for the deliverables. And DevOps is going to enable that type of things during the throughout the process because DevOps is going to give them enablers, give them a platform. The business folks can get into the development process, give them their feedbacks, and get the developers, based on their needs and based on requirements, and give them the right guidance.

So, one of the Agile principles says that business folks has to be part of the development process. To make that happen, DevOps is making a big chance, being enablers so that business folks can be part of it. The other thing it says, you mentioned that the customers can be part of the development process. How is possible? During the Agile process, it's really hard to get IT enabled environments and customer gives some feedback. Customers see what's really going on, what the release so far has been done. And they can give the right feedback to developers.

So, DevOps enabled that they gain, and the customers can give the right feedback straight away, and also the operational team, and get able to get a pass into the developers. That makes it much easier for the customers, make much easier for developers because they know what they're thinking about it.

Polling Question 2



Polling Question 2

Would you like more information about the relationship between Agile software development and DevOps?

1. Yes
2. No

Software Engineering Institute | Carnegie Mellon University
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University
9

**009 Shane McGraw: Okay so, we're going to a polling question. Folks, we are aware, there was a technical issue at the start of the event where I think we missed the first two slides of what Aaron went over. So, we are going-- Aaron, unless you can recap maybe quickly what you kicked off with, then we can go back into this polling question. But I know time's an issue to get through everything. But there was a technical issue that a lot of our audience couldn't see the speakers, couldn't hear audio. So, can you give us just a quick recap what was started at first? And then we'll--

C. Aaron Cois: Sure, no problem. So, the first few slides, what we were putting forth is the first misconception about DevOps, that misconception being that it's a fad, that it's a flash in the pan concept.

When in fact, recent surveys from Puppet Labs and ThoughWorks, industry leaders in the field, showed the organizations employing DevOps are deploying thirty times more often with fifty percent less failure. So, those kind of numbers are really hard to ignore. They show a massive amount of business value and show that DevOps is here to stay and cannot be ignored.

Shane McGraw: Okay so, we're going to bypass this polling question, issue of time. We'll get moving on. And if we have time at the end of the presentation, we can maybe give more deep about .

DevOps is not

DevOps is not
TOOLING

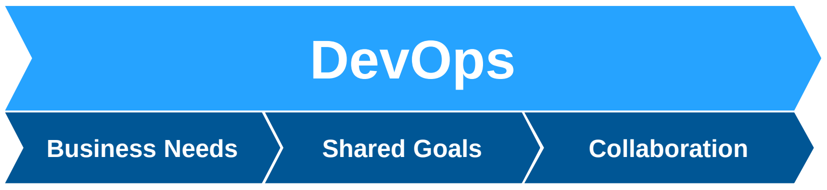
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

10

**010 C. Aaron Cois: That's no problem. So, the third misconception we wanted to talk about is the relationship between DevOps and tools. A lot of people, and now that the industry is growing up, there's a lot of work in the space developing great tools for DevOps. And we don't want to take away from that. But there is a common misconception that DevOps is just tooling, that it is all about adding tools, adding automation to your workflow, adding automation to your teams, that that is going to bring DevOps to your organization.

DevOps



Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

11

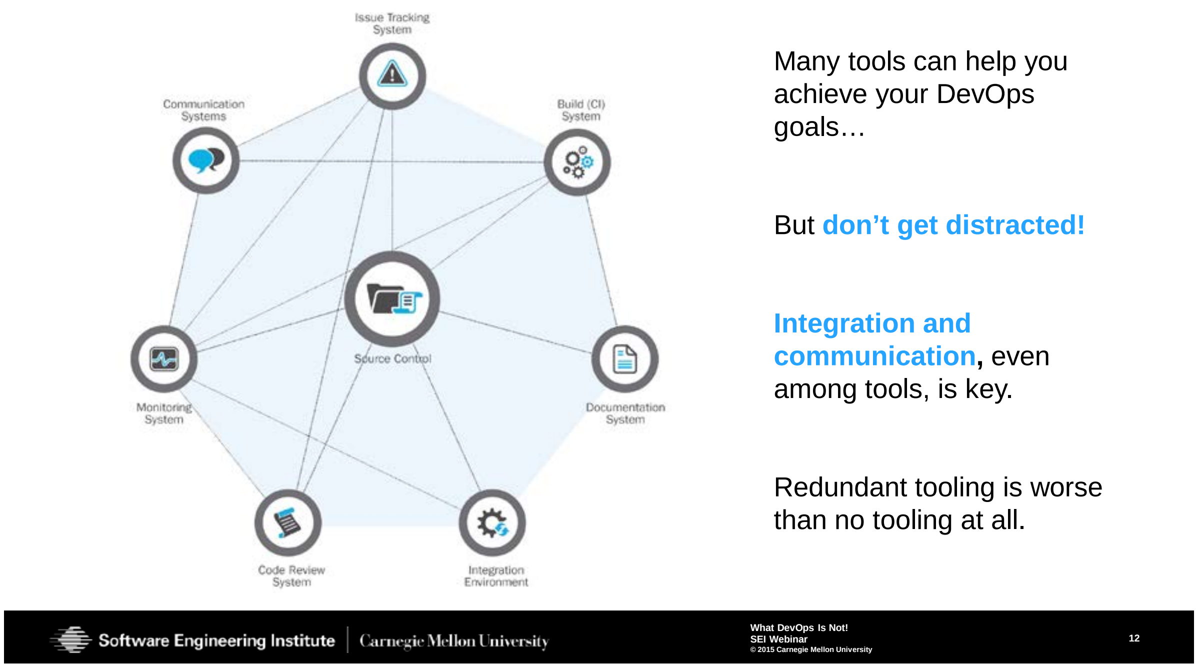
**011 And the reality is that DevOps is not about tools. Tools can help. But it is not fundamentally about tools. And tools do not make the DevOps.

DevOps is fundamentally built on a foundation of focus on business needs, shared goals between all members of your organization.

Everybody working in any way on a software product should be focused on the same set of goals, which is getting that software product out into the hands of users and providing value to your business.

If everybody doesn't have the same set of goals, you have risk. And collaboration between all those units once they have shared goals, they can collaborate easily and organically to meet those business needs. Those are the three concepts that are at the foundation of DevOps. And insofar as tools can help--

Many tools can help you achieve your DevOps goals…



Many tools can help you achieve your DevOps goals…

But **don't get distracted!**

Integration and communication, even among tools, is key.

Redundant tooling is worse than no tooling at all.

**012 Tools are great. But just bringing in tools will not create DevOps in your organization. What you can see here on this slide is the basic model of the types of tools we might see in a DevOps environment.

And again, we're not trying to say that tools cannot be extremely beneficial to an organization. But what's more important in these slots that you see on my generalized model of DevOps tooling is not the specific tools that go into each of those compartments, but the lines connecting them, which shows interaction. These tools need to be there purposefully. They need to be driving your organizational goals. And they need to be interacting in a way so as to establish transparency and communication throughout your process and throughout your organization.

Redundant tooling is one of the things we tend to see a lot in organizations. And this can actually be worse than having no tooling at all because it breaks down collaboration and shared goals if different departments, different teams in your organization, are using different tools to accomplish the same task, different tools that can't communicate with each other and can't share data.

Hasan, do you have any experience to add?

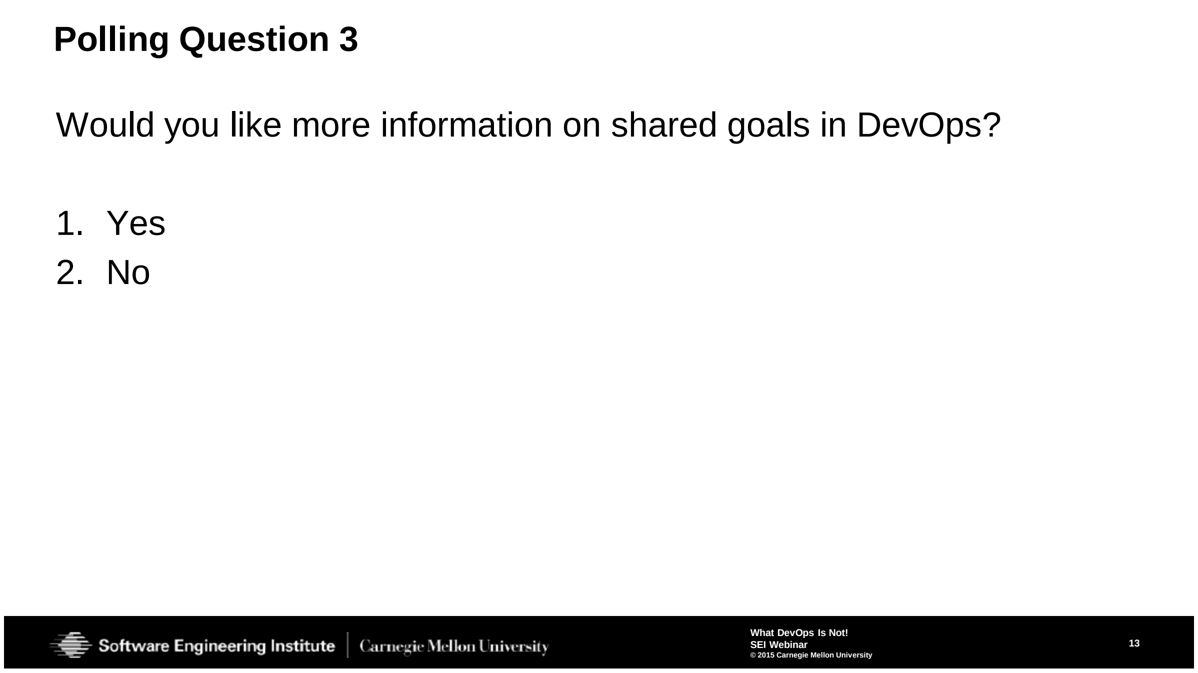
Hasan Yasar: I have a lot of experience actually based on my twenty years of experience. What I've seen so far, especially for enterprise companies, if you look at the large scale development things, and everybody is really focusing on within that organization as therefore and having their own development.

So, it's all on their own toolsets. However, they are not sharing their information between them. Like an example, if they're going to share the code throughout the email, it is not a good way to share the email because the code through email. They should have some sort of code repository platform where they can share the code with each other.

The other things also, we see that when they are communicate to the business folks like looking for the project management type things, looking for the tracker, looking for the deployment, looking for any deliverables, if the project manager has different tools and developer has a different tool, if they're not talking to each other, basically are wasting time. And developers are entering in two places. Or the project manager is coming and talking to developers, "Where's the status of the application?" So, it's creating a lot of time consuming throughout the business organizations.

Then I would like to say another thing as well. As a culture, I think DevOps is keeping the culture first. And tooling is needed. But it's not the first priority. The first priority is really people, which is the team, and also the cultural things. Tools is going to come after that. If you put the prioritization of tools first, you're going to lose the focus because to help, if anybody's going to use the tool, it's going to have. Otherwise, if nobody's going to use the tool, if they're not sharing with each other, the tool is going to be something that stayed by itself, which it doesn't solve the problem, basically.

Polling Question 3



Polling Question 3

Would you like more information on shared goals in DevOps?

1. Yes
2. No

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

13

**013 Shane McGraw: Okay, which is going to lead us to a quick polling question. The purpose of these polling questions, folks, is to get you an idea, make sure you have an understanding before we proceed to the next question. So, the polling question we'd like to ask now is would you like more information on shared goals in DevOps. And while you're voting, we can continue. We'll come back to the results.

C. Aaron Cois: Now, Hasan, you brought up a great point, a great point about a common bad scenario that we see. So, the example you used was a project management team needing a specific type of reporting, needing some specific data to then pass up to their management, pass up to the C-suite, and introducing a new tool to the developers, telling them to input information in there so that they could get the reporting they needed. Now, this is a common scenario. And this is a logical thing to see. Project management needs some specific information, needs reporting in a specific format to provide business value. Reporting does provide business value.

But developers have their own tools to deal with that sort of thing, issue trackers, and time trackers, commit logs. They are all very common tools that developers are already going to be using to track their work, to track their tasks. So, by introducing a new tool, now we've just increased the burden on the developers. We have increased the overhead and added a new potential source of user error because that information's being input independently into two separate system and may be inconsistent. We don't have any way of tracking that.

So, I would say that the DevOps solution to that problem, the right way to do it is to have the project management team come to the developers, discuss their needs, and say, "We need data in this reporting format. We need these type of reports to pass up to the executives to let them know status and to provide value to the business. And work with the developers to create some type of system that exports data from their existing toolset and puts into the reporting format the project managers need. That way you don't have that redundancy. Does that--?

Hasan Yasar: That's perfect. It makes sense because if you share the tools, and if you get enough-- the right tools together and sharing across a tool, that's flows off of the DevOps because developers are forced into sharing. If you're going to Share or if you use a similar toolset, then they communicate that. If I'm communicating something different, if you're communicating something different, if tools are forcing to deviate us what we are talking, that means we're not in the sync basically. I'm going spend my time. You're going to spend your time. And out of that, we're losing both times.

What really goal is get together in the same platform, whatever you are using it, let them talk each other, which is again the communication you're talking about. It's all the communications. DevOps is going to give them the communication. DevOps give them some sort of like using collaboration between the tools. Tools should be the second priority. I complete agree with you there.

C. Aaron Cois: Again, think of what, in that kind of example, what enabled that good solution. It was the project managers going and interacting with the developers, them having shared goals, and them communicating and collaborating to come up with a better, concise solution to the problem. That's really at the heart of this, that communication, that collaboration.

Shane McGraw: Okay. And back to our polling result. About seventy-five percent was looking for more information on shared goals.

C. Aaron Cois: Okay. So, we're going to try to hit that point as much as we can throughout the rest of this talk then because there is a lot to speak about when we talk about shared goals.

Hasan Yasar: Actually, I'm glad to hear that they're having a shared goals because shared goals are going to help us to really the main focus of the DevOps because we have to share goals means to share the goals as the business aspect of it, its quality aspect. There's a security aspect. There's a lot of things we can talk about it. As long as if the shared goals across the project team members, or virtually DevOps team members, is going to create a lot of good things and makes the project successful when they hear that.

C. Aaron Cois: All right, one of the things that comes to mind for me as a developer when I think about shared goals is I think about how often do we have a situation where my development team were working, were writing code, were building a

great system. And we build it. We
test it. We're ready to go. And we
hand it off to the production team, to
the release team, to the operations
team. And then we go out and
celebrate because we did our job.
That was our goal. That was our
deadline to create this system and
test it and have it ready.

And right there that concept-- and
we've all been there, the developers
are done. They've done their job.
They've proven that they've met their
requirements, and then they hand it
off to operations. I should not be
able to end my work right there.
We're losing shared goals by my
development team ending their work
there because the business hasn't
gotten value out of our work yet. The
business doesn't get value until it's
functioning, my software's
functioning in production, being used
by users. That's what generates
value and revenue for the business.

So, right there we see that my goal
as a software developer in a lot of
points in my career was not the same
as the goal of operations and was not
the same as the goal of the business.
And that's really what we mean when
we talk about shared goals.

Hasan Yasar: I agree.

Shane McGraw: Quick audience
question just because I know you
touched on this in the beginning when
we were having some issues. But Chitra
would like to know how DevOps is
different from an Agile framework?

C. Aaron Cois: So, that's a great question. Briefly, I think that DevOps really takes the concepts of Agile and applies it to other areas of the SDLC. So, Agile really focuses on the development process, on requirements, on integrating the customer into the Agile framework, and in some cases integrating business. But that's been something that Agile methodologies have had a hard time doing is getting business folks involved in the process as well. DevOps expands into sustainment, expands back into business, into security, legal, and talks about identifying what the expertise you need, what the quality attributes you need within your system are, and getting that expertise embedded, and then of course adds the layers of automation on top of that to smooth out the process.

Hasan Yasar: I look at a couple other things as well. So, when we talk about Agile we said DevOps is an extension of Agile. It's not only a different thing from Agile. It is an extension. So, why that DevOps came into reality-- it's kind of like a technology is forcing us to use the DevOps methodologies because if we talk about Agile principles, it is looking for getting customer engagements, getting business engagements. These are the manifests-- one of the twelve principles of Agile. If you think about how we can get the customer involved into Agile process or get how business Agile process, it cannot be a pen on paper. It has to be some sort of automated way to get the customers or getting a business analyst or other stakeholders in the project, the development cycles, so they can really get the inputs throughout the process.

To that things, DevOps is going to help them because DevOps is automating the process. DevOps is helping the customers get into the set up that opens right away. Not really sitting in the meetings and daily Scrum meetings or daily stand-ups or the iterations, but having a DevOps in mind is going to create a lot of interaction between the developers, analysts, operational teams together so they can same towards the goals. So, it's not a different. It's really an extension of the world. It's kind of like a forcing.

And other things we have to talk about as well like new generations, technology, is that developments or the micro-scale architectural type of things. It is really forcing have a modular approach. That means we have to have a continuous deployment. That means you have to get customer response as quickly as possible, get the user response and feedback as quickly as possible so you can get your business increased because it's kind of like a race against other companies. So, if you miss that race there's really hard to catch up later on.

DevOps is not

DevOps is not

A PRODUCT

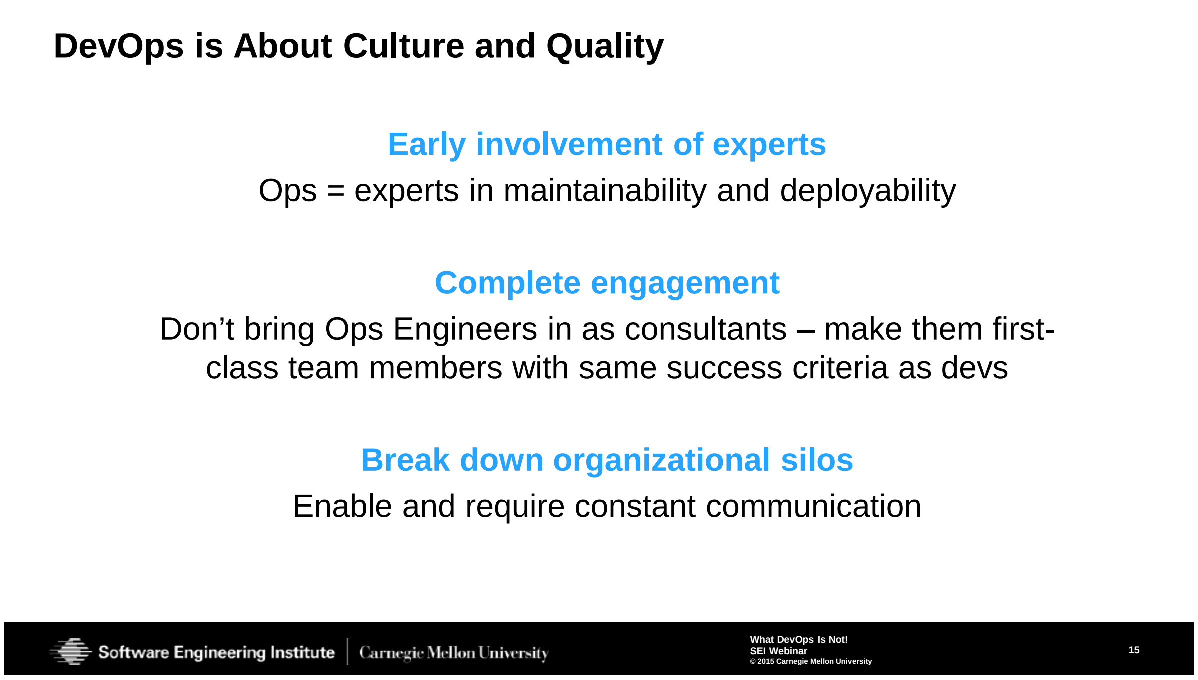
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

14

**014 C. Aaron Cois: Right. I think we'll talk concepts touching on that as we go forward. And so the next misconception, what DevOps is not. DevOps is not a product. There will be a lot of companies coming up that may try to sell you DevOps. And some people may be enticed to try to just buy a product, push it into their environment and say, "All right, now we have DevOps."

DevOps is About Culture and Quality



DevOps is About Culture and Quality

Early involvement of experts

Ops = experts in maintainability and deployability

Complete engagement

Don’t bring Ops Engineers in as consultants – make them first-class team members with same success criteria as devs

Break down organizational silos

Enable and require constant communication

Software Engineering Institute | Carnegie Mellon University

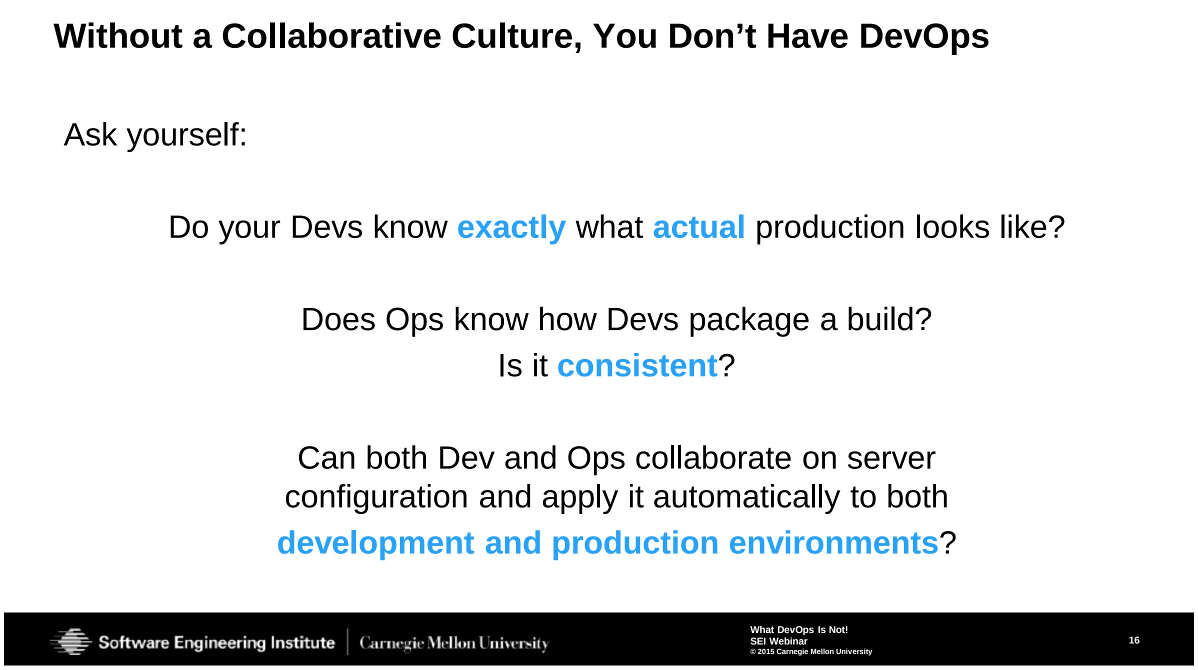
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

15

**015 Or even hire someone, and that's hiring DevOps. But DevOps isn't really a product. It's not something you can buy. DevOps is fundamentally about culture and about the quality of your application. And by quality I mean the specific software engineering term of quality, of different quality attributes. What matters to you? What do you need? Do you need high scalability? Do you need high resiliency? Do you need high usability? Different projects, different companies focus on different levels of quality to meet their market needs. But DevOps is about that and it's about the culture within your organization, the collaborative culture, the communication culture.

It's about getting experts in whatever quality you need. If it's maintainability and deployability, that's getting your ops group embedded in your projects. And it's bringing them in as full, engaged members of the project team, giving them ownership of the project just as the devs have ownership. Not just bringing in an ops engineer to consult every few months on the project to make sure it's going in the right place because that doesn't give you the real-time feedback that we're looking for. Ops has to be there to answer any questions or to speak up when a feature's being discussed that the developers won't even know is going to impact production, to speak up and say, "Hey, that will impact production because our environment is like this. Don't go down that path." That is the kind of real-time feedback that is going to keep development on track and is going to give you the benefits of speed, and efficiency, and effectiveness that DevOps promises.

Without a Collaborative Culture, You Don’t Have DevOps



Without a Collaborative Culture, You Don’t Have DevOps

Ask yourself:

Do your Devs know **exactly** what **actual** production looks like?

Does Ops know how Devs package a build?
Is it **consistent**?

Can both Dev and Ops collaborate on server configuration and apply it automatically to both **development and production environments**?

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

16

**016 So, what you really have to ask yourself when you're talking about DevOps, again, it's not about a product. It's about what you know and how your company is structured, how your team works together. So, one of the questions I like to ask is, "Do your developers-- does your development team know exactly what actual production looks like. I mean exactly. Can they tell you what production looks like?" Because if they can't, if they don't know what production looks like, how can they design software for it? How can they make sure that the product that they put out is going to meet the business needs is going to work in production for users? It's obvious, but it's also a very common disconnect.

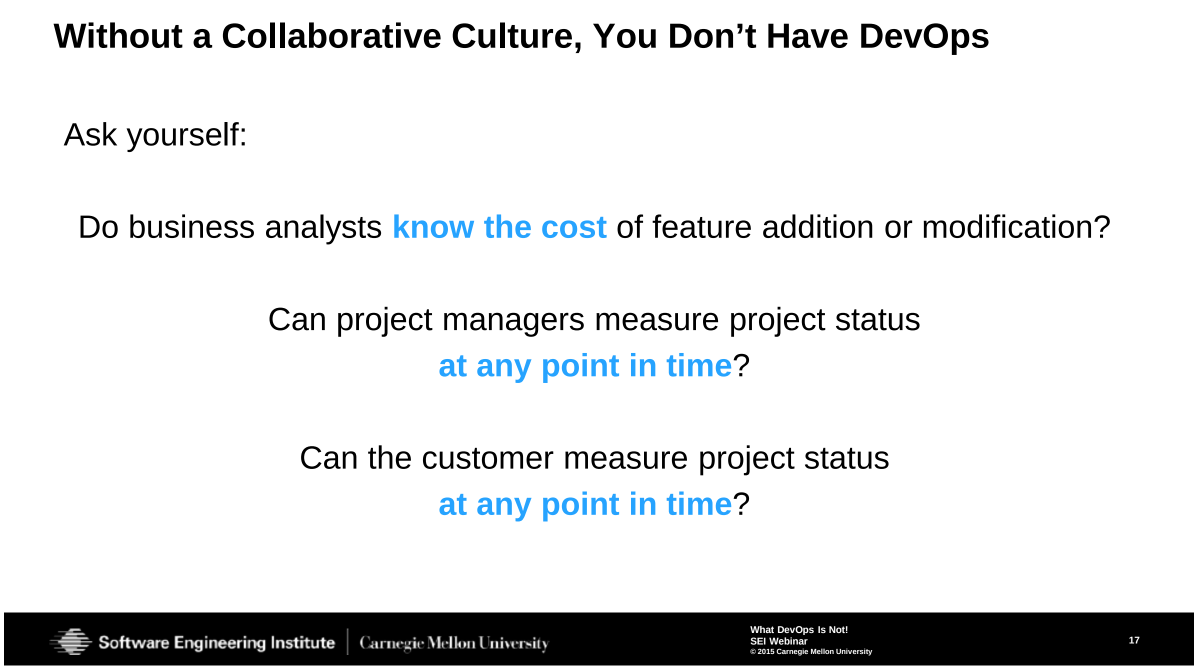
Hasan, do you want to go through something more of these?

Hasan Yasar: Actually, we have a lot of things we can cover up on the way because now we can talk about all the silo stuff. And we can talk about other topics, which is the operational teams, how they can get together, how the other team came together so we can add more on down the road. Is there any questions, Shane?

Shane McGraw: There is. One from Christopher asking do you feel there's an intersection between continuous integration and DevOps.

C. Aaron Cois: I can try that, because absolutely. Continuous integration is a fundamental piece of the DevOps puzzle. It is your gatekeeper. We'll talk a little bit more about this in a later part of this presentation but I think continuous integration is absolutely pivotal in reinforcing the types of quality that you want to achieve. Continuous integration is what's going to enforce the rules that you want on your product before it goes forward into staging or into productions. So, and this is going to be customized to what your business needs, again, what type of quality do you need.

Without a Collaborative Culture, You Don't Have DevOps



Without a Collaborative Culture, You Don't Have DevOps

Ask yourself:

Do business analysts **know the cost** of feature addition or modification?

Can project managers measure project status **at any point in time**?

Can the customer measure project status **at any point in time**?

Software Engineering Institute | Carnegie Mellon University
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University
17

**017 But it's a very, very important means of measurement and also of enforcement of your process and of your DevOps.

Shane McGraw: And can we squeeze in one more just while we're on this section from Matt who's asking how is DevOps done in a regulated industry like biotech, which is risk averse space?

So, that's a very interesting question. That's a common question is when you get into an industry that's--biotech is one, financial is another--heavily, heavily regulated, how do you deal with DevOps. Well, a lot of the concepts and why we say that you really have to focus on culture, and on your process rather than on specific tools or specific technologies is partly for that reason. Right? However you achieve the types of

communication and collaboration that
are the hallmark of what DevOps is
what's going to work for you.

Obviously, I will achieve those goals
differently in a small, Silicon Valley
web startup than I might in a highly-
regulated biotech company. But the
goal should be the same, identifying
what your business needs, trying to
find where the bottlenecks are in
getting my work through my process
and into production, and then seeing
ways to remove those bottlenecks, to
get things through faster, and to
make sure that all the information, all
the expertise, is known early in the
process. So, in a regulated industry, I
would say that one of your most
important stakeholder expertise sets
is legal. Right?

Hasan Yasar: I think it's going to
DevOps really help them a lot
because if they use as expertise in
development cycles like bank
industries or biomedical, any
of these industries, if they have the
core attributes or the goals they
define at the beginning, so
developers know what they're dealing
about it instead of going at them and
changing. One of the things we
should remember is that DevOps is really
changing the thinking and get
everybody onboard if they're in the
lifecycle, the project lifecycle. See, if
you think about that one, really a
quality attribute at end of your
development cycles. See, it can be
like a compliance check. It can be
another privacy. It can be another
quality attribute. So, that can be part

of the overall process, which is DevOps is helping them because DevOps mentality says get expert into the project team. Project team's mutual.

C. Aaron Cois: Right. So, in that case it's your regulatory affairs professions. Right?

Hasan Yasar: Absolutely, it's going to give them a lot actually. So, that's one of the things they should involve instead of having a siloed, instead of legal aspect or other stakeholders, they cannot do by themselves and give them some sort of like a guidelines. But developers, instead of spending time to read everything else, but if one body is going to be in that meetings during the development cycles, they will have a lot of input during the development process. And then end of that ever going to address the needs during the process including integration, including deployment. There's many ways to step in the process why they can address the needs throughout the process basically. It's really helping DevOps.

Shane McGraw: Right.

DevOps is not

DevOps is not

ONE SIZE FITS ALL

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

18

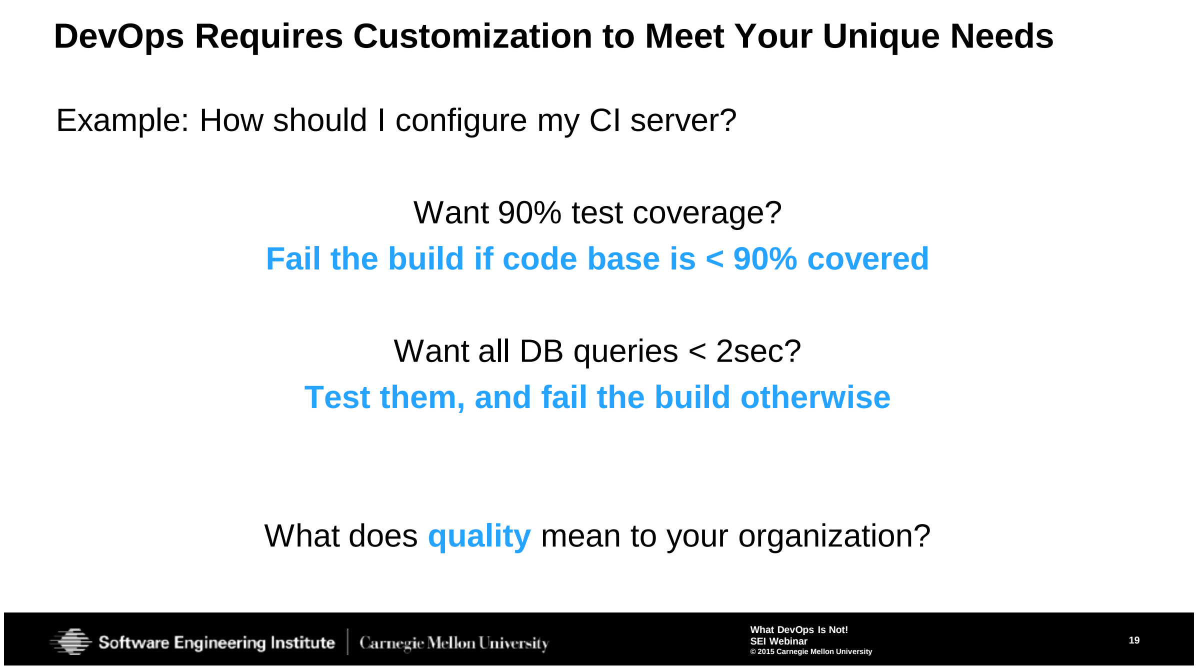
**018 Exactly, it may look different, but it's still DevOps.

Hasan Yasar: Absolutely.

Shane McGraw: Getting those regulatory professionals in there early and making sure you meet your regulatory needs is fundamental to your business in a regulated environment.

So, the next thing, moving on, that DevOps is not. DevOps is not a one size fits all solution.

DevOps Requires Customization to Meet Your Unique Needs



DevOps Requires Customization to Meet Your Unique Needs

Example: How should I configure my CI server?

Want 90% test coverage?
Fail the build if code base is < 90% covered

Want all DB queries < 2sec?
Test them, and fail the build otherwise

What does **quality** mean to your organization?

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

19

**019 DevOps needs to be customized to your organization to meet your needs. And we've touched on this a little bit before. The question about continuous integration was greatly foreshadowing of this. How I organize my process, how I implement DevOps in my organization needs to be specific to what my organization's needs are.

So, as a small example-- and CI is one of my great go-to technologies for this. If something that's very important to my organization for whatever reason, that provides my business value, is that I want high code coverage, high code test coverage-- Say, if I want ninety percent test coverage, and I want to ensure that I am keeping that level of quality throughout my entire process, then I'm going to configure my continuous integration server to fail

the build any time code coverage
isn't met. All my other tests can pass.
My software can build. It can work.
But if test coverage isn't there and that's
important to my business, the build
fails, developers have to go back and
make sure that they meet that goal.
It's a way of incentivizing. It's
behavioral economics.

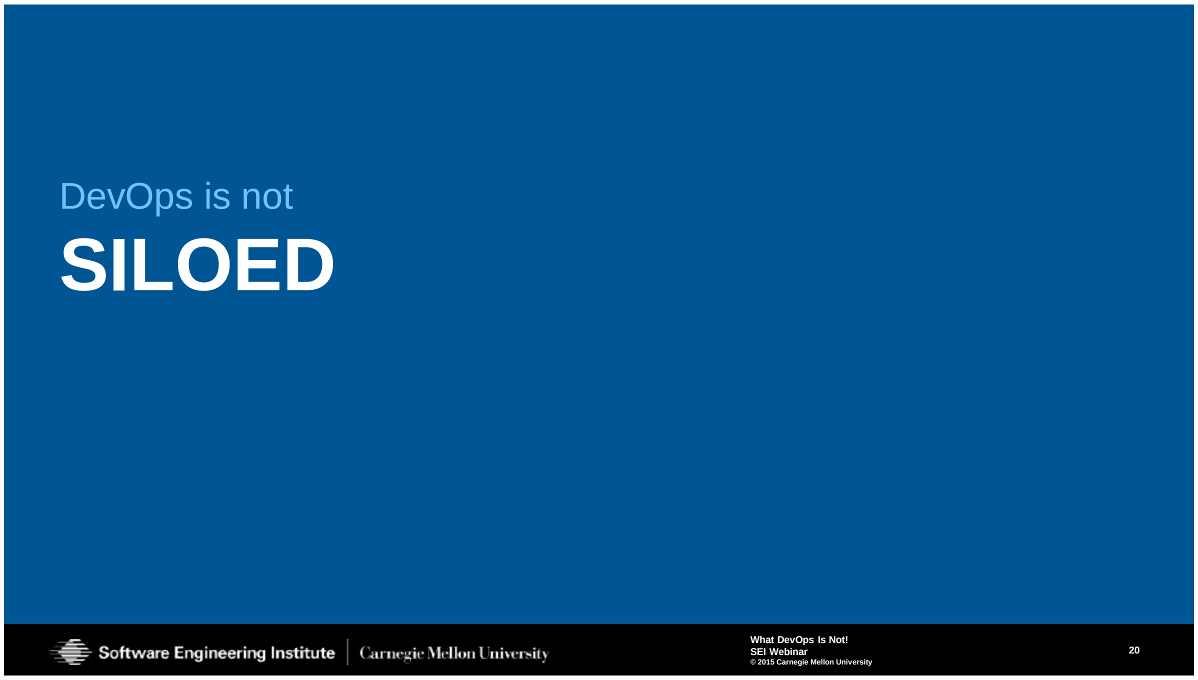
If performance is important, if we
need a highly scalable storage
solution, and we even need to make
sure that all of our database queries
are happening in highly performant
way, test that in continuous
integration. Fail the build if the
performance isn't up to snuff. Again,
code could compile. Tests don't pass.
But what quality is important? What
does it mean before I will release
something into staging?

Hasan Yasar: Other things actually
can add in on top of quality. If the
organization is a really small
organization and or the project size is
small, they don't need to deploy
every minute or every hour. They can
set the cycles. They can set the
project expectation. They can-- they
know the user needs. So, instead of
modeling how Amazon does, how
other companies does, they can
define something for their own
business needs. So, they can really
deploy depends on user base,
depends on their critical needs. So,
instead of really modeling exactly I
have to do ten times in a day. No,
you don't need it ten to ten times a
day. You need to have to have
maybe a month's, maybe twice a

year as long as what the business needs, your needs. So, that they can define your process basically.

C. Aaron Cois: Right. And so, that's the fundamental question, what does quality mean to your organization.

DevOps is not



DevOps is not

SILOED

Software Engineering Institute | Carnegie Mellon University

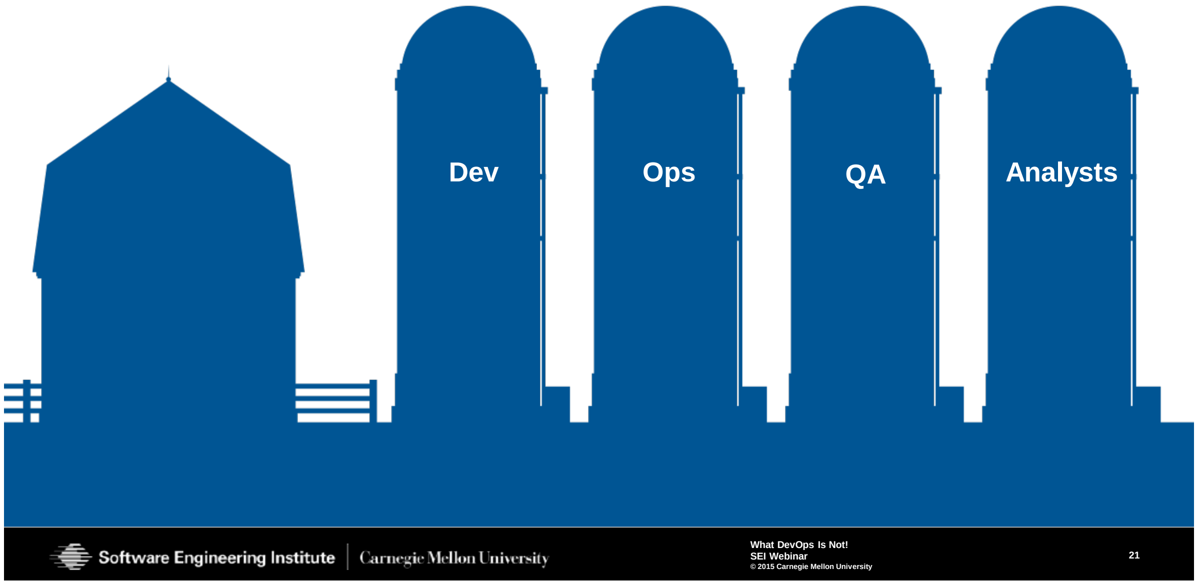
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

20

**020 What bar do you need to achieve? And then model your DevOps based on that.

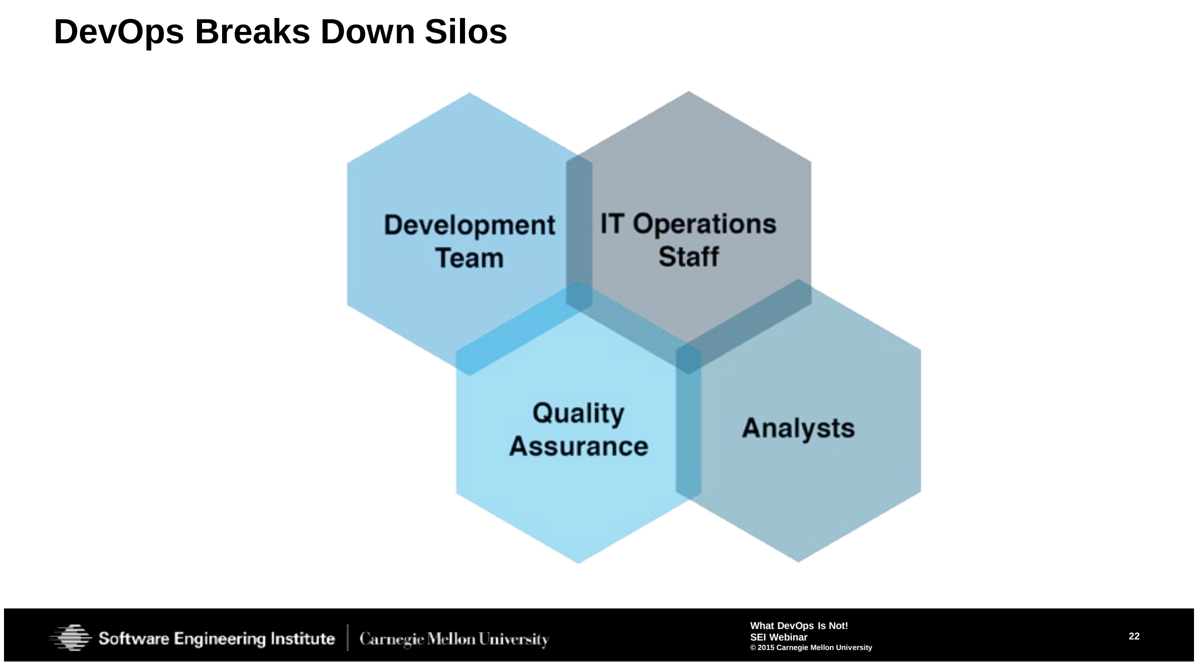
So, another thing that DevOps is not, and this is a fundamental principle that DevOps is trying to bring to software development, DevOps is not siloed.

Dev


Dev
Ops
QA
Analysts
Software Engineering Institute | Carnegie Mellon University
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University
21

**021 Many organizations are going to look something like this. We've all been in organizations, many of us probably still are in organizations where you see a development department, an operations department, a quality assurance department, a department full of business analysts. And these are silos within the organization. DevOps is fundamentally working against silos, trying to break those silos down because those silos impact the things we've been talking about. They're going to inhibit communications and inhibit collaboration.

DevOps Breaks Down Silos


Development Team
IT Operations Staff
Quality Assurance
Analysts

DevOps Breaks Down Silos

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

22

**022 We're not saying that we want to break down the entire organizational structure and make every organization a flat organization, redefine everything, wipe out the management chains. All we're saying is that these siloed departments need to have free and open communication between them. They need to be brought together to have visibility between their internal operations so that all of the staff members can organically and easily collaborate between each other.

I know for myself some of the best teams I've worked on, some of the best projects I've seen, have been great because it just so happened that the developers, and the operations staff, and the quality assurance staff that were working on those projects knew each other and trusted each other, and just naturally went to each other with questions to sort out issues without having to be ordered from their management chain to do so and run up through the management chain and back down to get answers. They just organically worked together and that made things happen faster and more efficiently. Have you seen similar things?

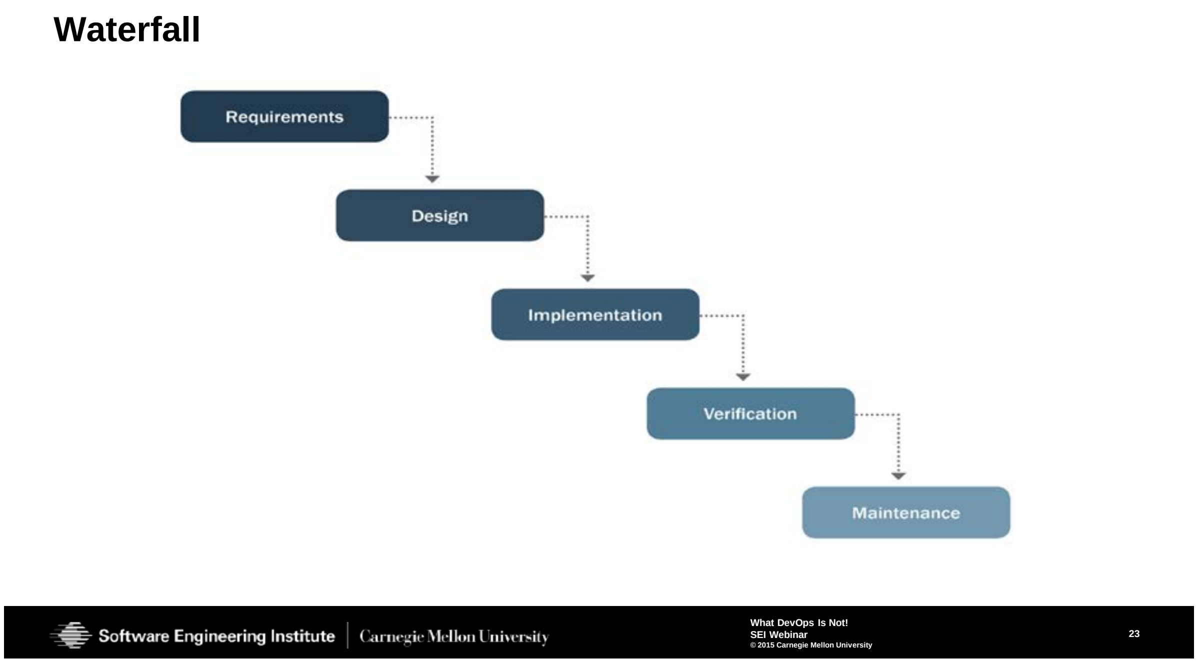
Hasan Yasar: I see a lot of things actually, especially in the enterprise organization. There's a lot of bureaucracy, right, bureaucracy of the management chain. If the creative people are not able to communicate with developers in a regular basis, or the business analysts, they have their own process. And the developers use-- they have to connect with the developers. It's totally creating a lot of siloed environments. The DevOps is going to tell them to bring together and share the same goals. Goal is always the business needs. So, as long as they share the goals, then that's the things we're going to get together and break the whole siloed environment so they can really work as a group together and share their feedbacks.

C. Aaron Cois: Right. And that brings that real-time feedback in like you just said.

Hasan Yasar: That's a critical point there.

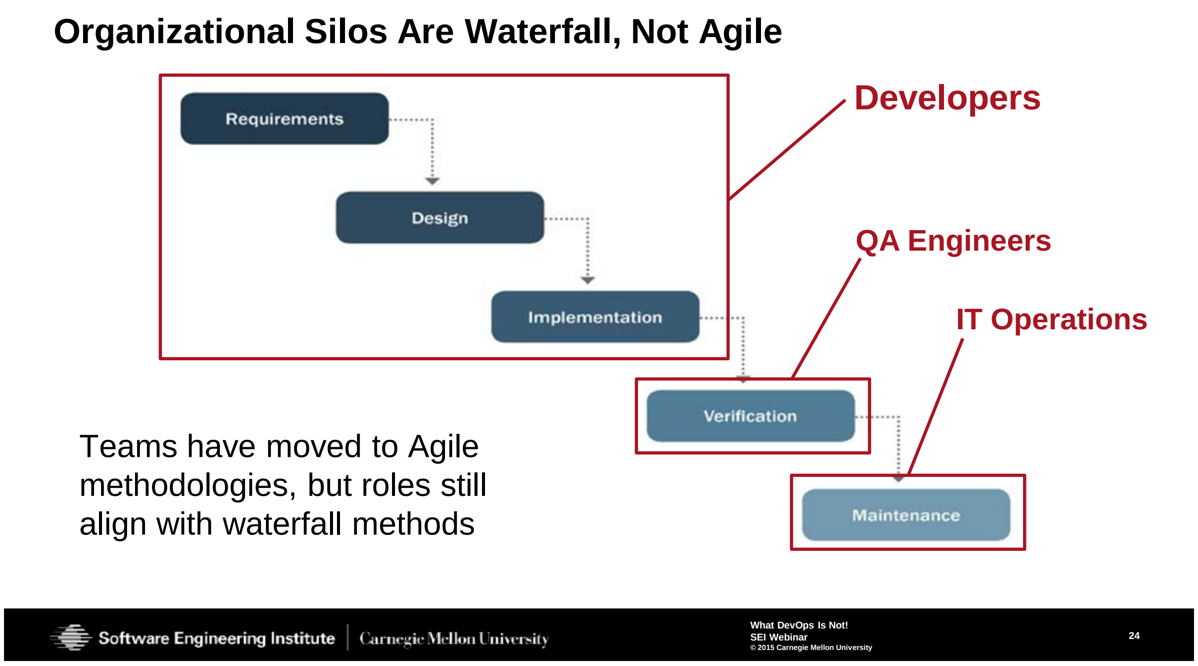
C. Aaron Cois: Real-time feedback. If I have to ask a question of my operations unit, and that requires me to pass that question up to my manager who passes it over to the manager of the operations unit who passes it down to the appropriate operations staffer, and then the answer comes back up through that path, that takes a long time. That takes days in some organizations maybe even weeks for me to get a simple answer. Well, what happened in that time? I may have not been able to work on that feature because I really needed that answer, or I may have gone down a path that was misguided because I didn't have the feedback that I needed. As a developer, that has wasted time in one way or another.

Waterfall



**023 Now, what does that do over the course of the project? Well, those bits of wasted time add up. And they slow our feature development. Now, this concept of silos-- and we see in a lot of organizations-- and interesting note, let's just take a quick look at the waterfall methodology, the methodology that Agile has been trying to reduce in operations. So, waterfall says we go from requirements. We do all the requirements. Then we go to design, do all the design. Finish it. Then we go to implementation. We never go backwards. No changes. No adjustments. Then we go to verification. Then we go to maintenance. Well--

Organizational Silos Are Waterfall, Not Agile



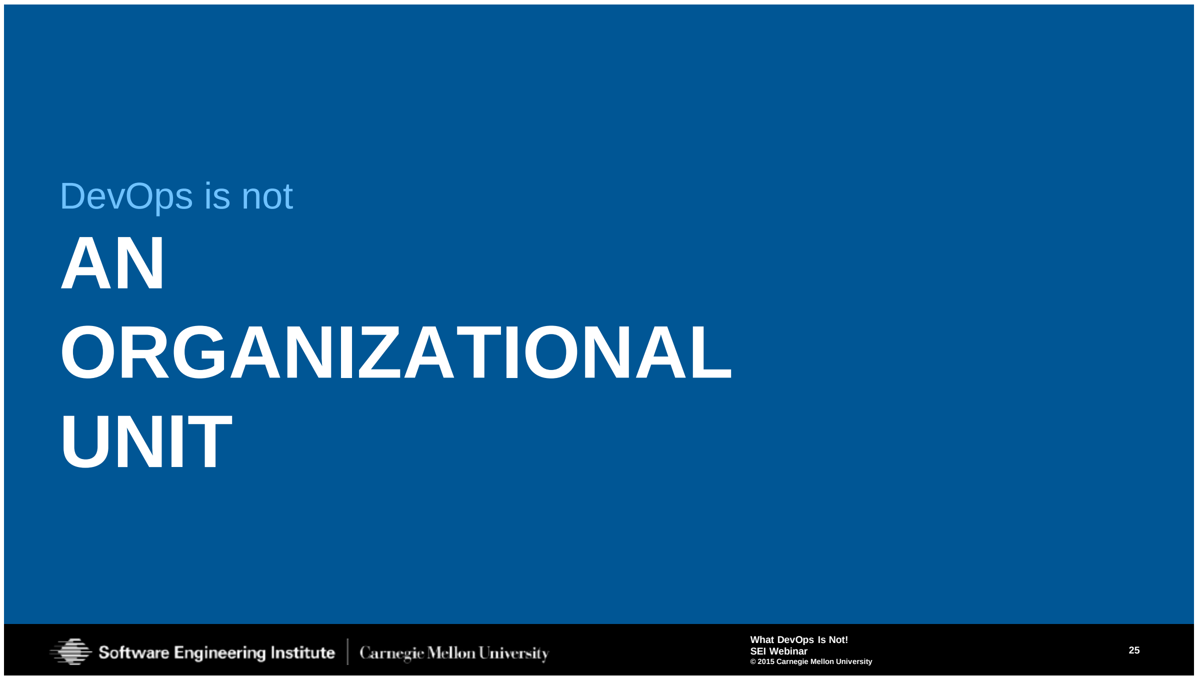
Organizational Silos Are Waterfall, Not Agile

Teams have moved to Agile methodologies, but roles still align with waterfall methods

**024 This process, if you lay on top of an organizational silos, you can see where the influence came from. Waterfall methodologies reinforced the way that silos have been determined and constructed within an organization. So, in fact a siloed organization that doesn't have good communication and collaboration flow between their groups is actually going to be highly aligned with the waterfall methodology. They're going to have problems implementing Agile and more problems implementing DevOps.

The communication structure can be highly defined by the organizational structure. So, it is up to us in organizations to try to impact that culture and try to break down these barriers. And again, we're not saying that you don't still have separate departments and reporting chains but to try to make a free flow of information and communication to get away from waterfall methodologies.

AN ORGANIZATIONAL UNIT



DevOps is not

AN ORGANIZATIONAL UNIT

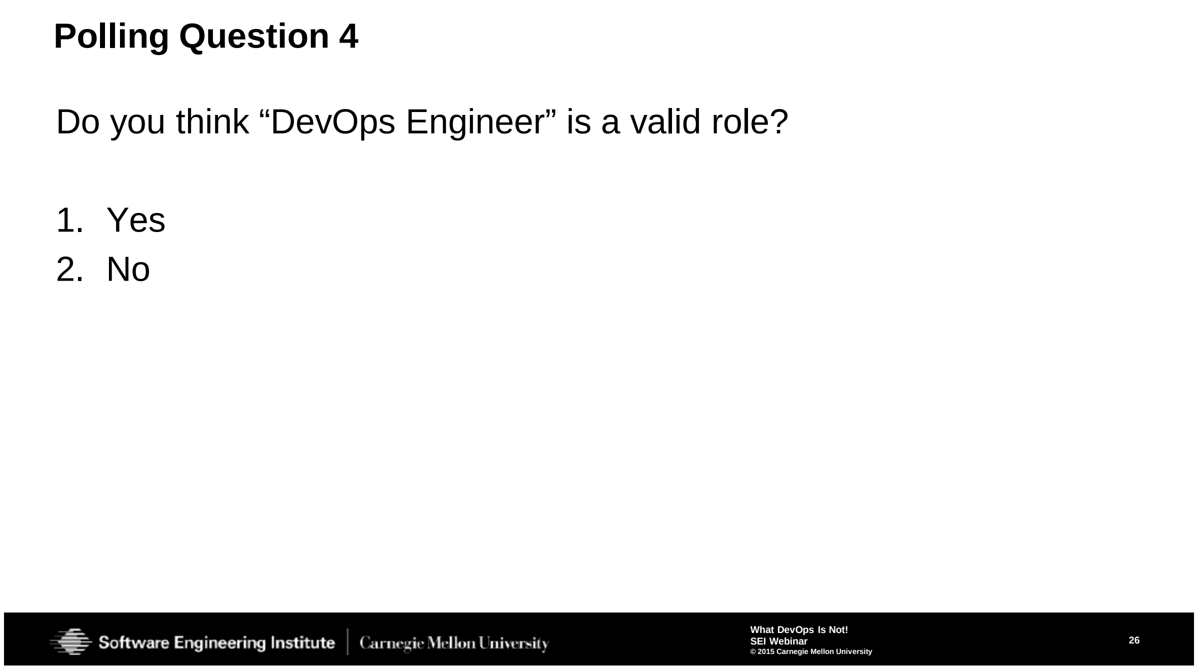
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

25

**025 One of the other things that we like to talk about as far as DevOps is concerned and a misconception we see in a lot of our clients is that DevOps is an organizational unit--

Polling Question 4


Polling Question 4

Do you think “DevOps Engineer” is a valid role?

1. Yes
2. No

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

26

**026 That DevOps is a role. So, at this point I'm going to turn it over to Shane for a quick polling question to get where you guys stand on that.

Shane McGraw: Okay. Our next polling question we'd like to know is do you think DevOps engineer is a valid role. And while we give the audience a minute or thirty seconds to vote, we can throw in a question here real quick?

C. Aaron Cois: Yeah, absolutely.

Shane McGraw: Okay. Leslie wanted to know how does DevOps embrace the integration of large projects, in particular projects involving hardware. It seems as if Agile does not embrace this.

Hasan Yasar: Actually, my background is a double EE, as an

electronic engineer. It's really a
challenging topic getting your
hardware into the software
development process. So, if you look
at for the software development is a
lot of micro applications. A lot of
things can be modular. However,
maybe twenty years ago, it was really
difficult to get the hardware as part
of the development process. In these
technologies, it's really getting easier
and easier.

So, like one example is really can get
the hardware put in a simulated platform
for. If you know what's in and out,
there is no hardware that's working
by itself. It is looking some sort of
software integration including
firmware, including other API sets,
including other communication portal
stuff. So, as long as the hardware
unit is defined as the simulated
module by itself, it's self-contained,
DevOps can't help up either because
if the self-contained hardware is
going to be part of as a software--
So, if the hardware guys know what's
in and out, while they were working
on improving the quality or improving
other part of their design everything
else and the back end, but if the
software guys know what is the
interaction between the hardware
unit, so they can probably get talk.
And then they can see each other.
They can really get integrate much
quicker than versus waiting the
hardware ready, and prototype ready
so they can get the prototype. And
the firm is going to push it to the
prototype and then other testing. So,
if the hardware is defined as

simulated platform overall is it's much easier to get involved at earliest stage which is possible actually. So, I think about is how can I do the simulated platform my hardware unit as a simulator so I can really integrate each other. It's a key point.

C. Aaron Cois: Right. I agree completely. It's about simulation and again about real-time collaboration. So, having that simulation platform and then enabling real-time collaboration between your hardware and your software folks, if the hardware people, as they're implementing new features can immediately update the simulation platform, then you have that real-time back and forth. Then you can move forward.

Hasan Yasar: See, you are going back to the same thing like changing the cultural things actually.

C. Aaron Cois: Yes.

Hasan Yasar: Changing total philosophies. I don't want to get just my heart of an engineer. I'm touching some software. But software engineering's also doing some software development. So, if they're going to get together and talk, if part of what an engineer knows is what is my input and output for just a basic definition, so if they're able to simulate some of the hardware component, then software guys can interact each other.

C. Aaron Cois: Right. That simulation provides the interface for them to collaborate and communicate on. And that's what you're looking for. And again, to reinforce that point, when we say communication and collaboration, we don't just mean everybody always has to be in meetings together. We mean the opposite. It's finding those ways to collaborate in their technical work, which is the simulation platform.

Hasan Yasar: Basically sharing. So, typical way like I remember a long time ago when I started first in hardware development, it is just waiting before the last minute type of thing. So, I have to get my prototype. It took me maybe six months, maybe a year. And during that time, software is thinking it's going to work out. When we get together for the firmware and the hardware guy's looking for where's this other operational needs, it's going to create more problem. So, if I know at the early stage, as a hardware engineer, I can say, "Okay, I'm looking for some device that's working such and such conditions, power requirements, operations requirements and I can write the software based on the hardware requirements, which I should know at the beginning not at the end. So, thinking is able to bring all together as early as possible. That's the one of the philosophies, which is great for the hardware platform as well, which is-- and technology is making it much easier right now.

If you're looking for any circuit design application, they have a simulated platform. You can simulate that. If you look the mobile platform there is an emulator. You can get the emulator. You don't need the hardware right away. Get the emulator put in your machine and then you can emulate what hardware's going to react on. So, there's a certain way. It's not through a hundred percent is compatible but the certain way to do that. Just changing the thought philosophies, changing the cultural things makes much easier.

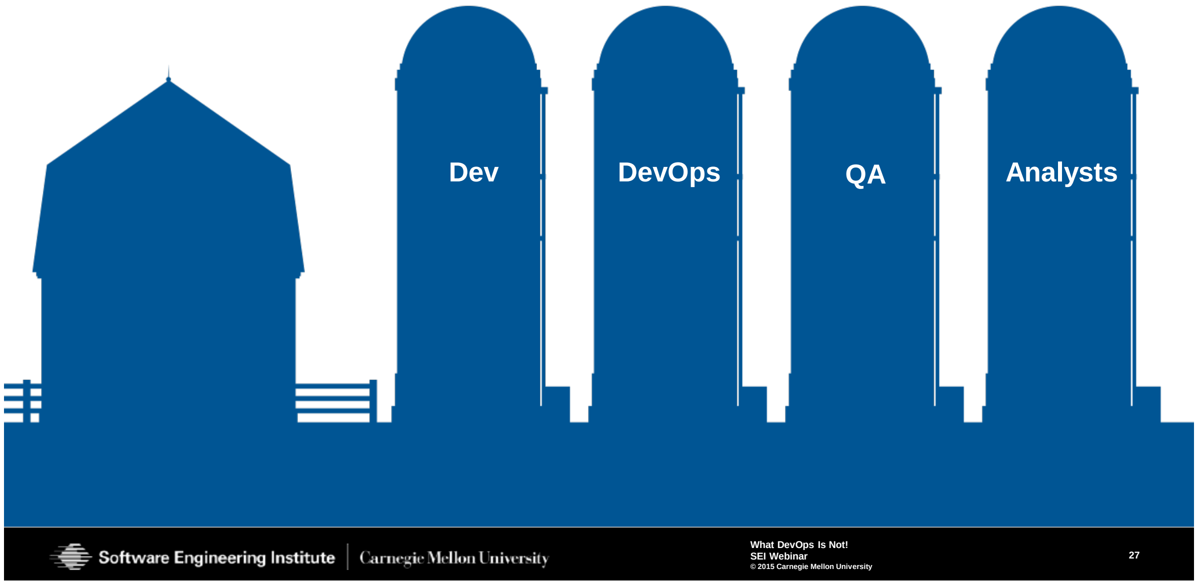
C. Aaron Cois: Right. You're not going to get away from real hardware testing for acceptance, but you can iterate on that simulation platform. Keep it as close to reality as possible.

Hasan Yasar: Correct.

Shane McGraw: Great. So, we'll share the results here now. And I believe the last I saw was fifty one to forty-nine percent that yes was the-- or actually it's fifty-fifty. We had a hundred and twenty-two responses and a fifty-fifty straight down the middle.

C. Aaron Cois: So, that's exactly what we expected. This is a hot topic. And there's a lot of diverging thoughts on this concept. And if you hit a job board right now, I guarantee you will pull up a ton of roles that are DevOps engineer.

Dev


Dev
DevOps
QA
Analysts
Software Engineering Institute | Carnegie Mellon University
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University
27

**027 Now, here's what's interesting. Read through those roles and try to decide what the actual background is for a DevOps engineer role. In some cases, it will read just like a sys admin that knows how to do a little software. In some cases, it will read like a developer plus a sys admin. In some cases it will read like a developer that knows how to do a little scripting.

So, fundamentally when we talk about DevOps being an organizational unit, I think that that is a misstep. I think that that is a worry because effectively you're looking at that same siloed model up here on the screen and replacing your ops silo with a DevOps silo. If you're just hiring people called DevOps engineer instead of operations engineer or a system administrator, you're not actually changing your organization's culture. You're just maybe expanding the role you expect certain people to play.

And we're seeing a lot of pushback in the developer community about developers being convinced that DevOps is just trying to get them to do two people's jobs, to do a devs job and an ops staff's job. And that's dangerous too because those are full-time, high expertise professions. You can't expect somebody to be an expert in both of those things. One of those things is going to fall by the wayside.

So, I would be a little worried if you start to see DevOps-- if you start to think about hiring a DevOps engineer and don't have a solid idea as to what the role you want that person to play is. I would say the better phrasing is to talk about a developer that is comfortable and experienced operating in a DevOps environment. The environment is the thing that is DevOps. Or a system administrator, an operational engineer that is comfortable operating in a DevOps environment. That's the role.

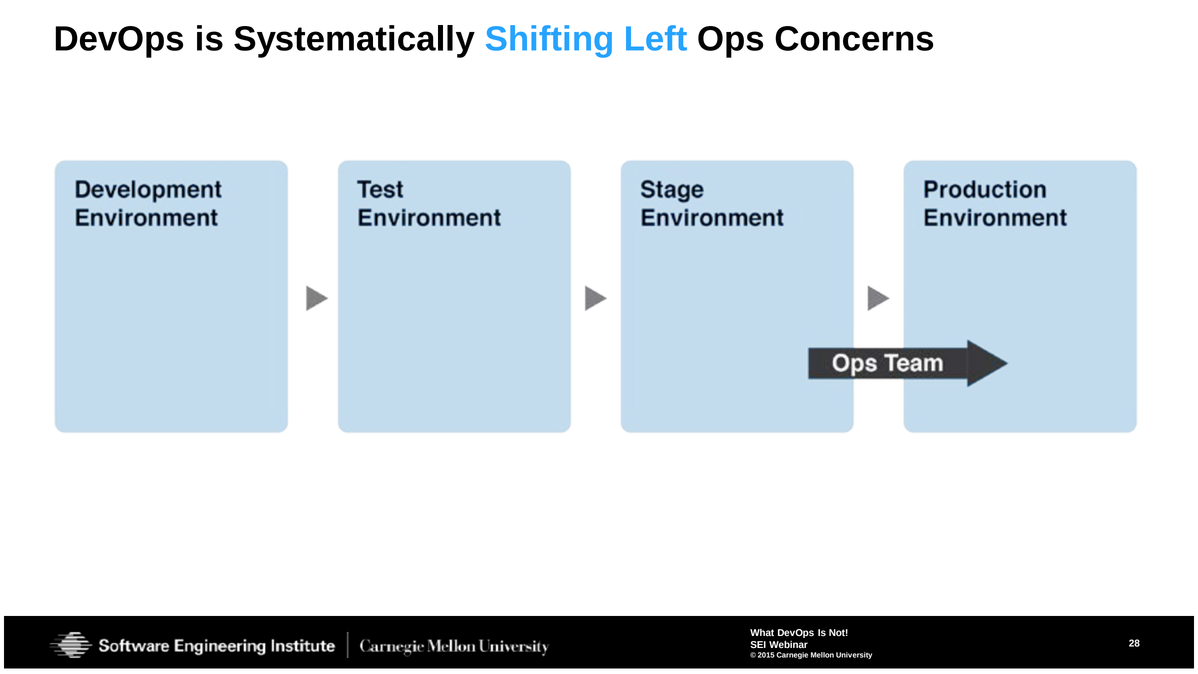
Hasan Yasar: We'll keep talking. We'll keep talking. We don't want to create a silo. If you're going to create a DevOps as a silo, as another team or engineering group, we're creating another silo. We're against that actually. We are saying we should have expertise together that knows enough that can get idea for other team members instead of really creating a separate unit as a DevOps unit.

C. Aaron Cois: Right.

Hasan Yasar: And then creating another silo, which is not good again for DevOps principles.

C. Aaron Cois: Because again, DevOps practices is about every stakeholder, about if you have an ops and information security and QA. So, they should all know how to work in a DevOps environment. So, all of them get that tag or nobody does.

DevOps is Systematically Shifting Left Ops Concerns


DevOps is Systematically Shifting Left Ops Concerns

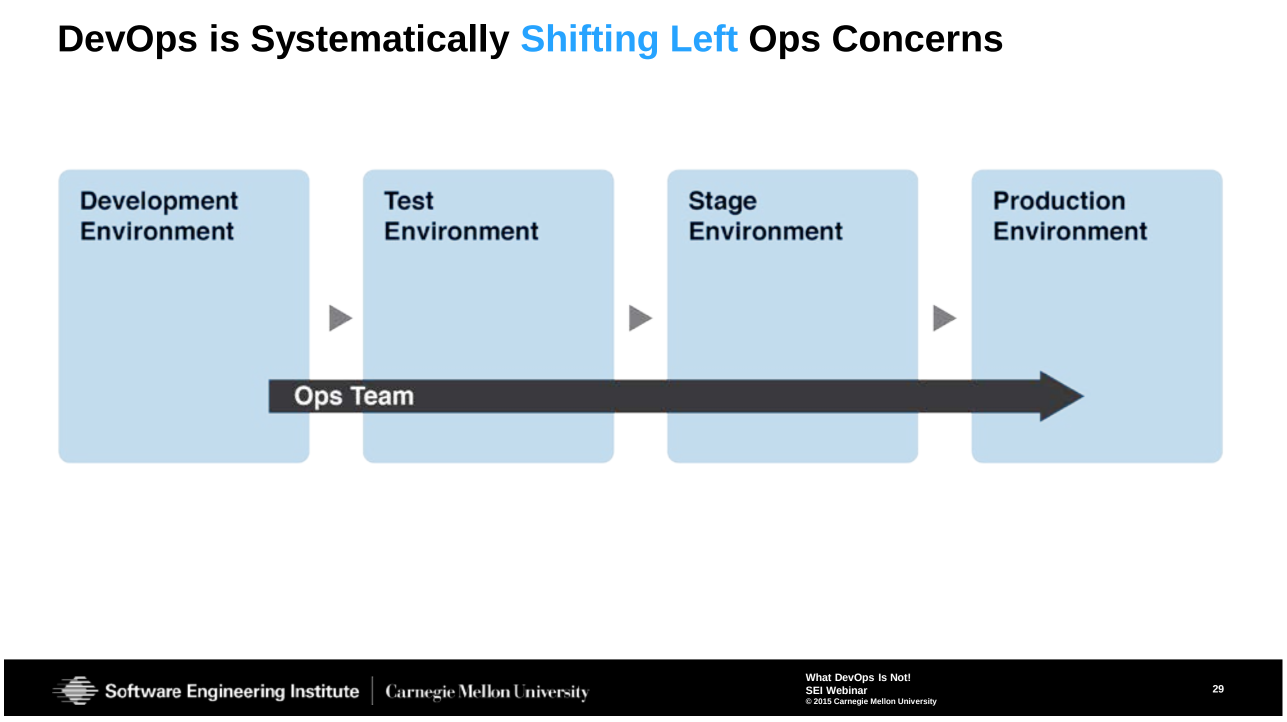
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

28

**028 Shane McGraw: So, we'll roll out the question we just got--

DevOps is Systematically Shifting Left Ops Concerns



DevOps is Systematically Shifting Left Ops Concerns

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

29

**029 From Allie saying wouldn't
DevOps increase cost.

DevOps is not

DevOps is not

JUST ABOUT DEV AND OPS

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

30

**030 C. Aaron Cois: Well, it depends on where your costs are coming from. So, implemented correctly and efficiently what DevOps is supposed to do is increase the feedback throughout your lifecycle. So, I guess my question would be where do you see those costs coming in from. A lot of people think that I have to hire a new department called the DevOps department. Okay, if you hired a new silo, that would increase costs. But that's not what we're thinking you should do.

The other thing they think as well, my operations engineers have to be embedded in the project full-time, all the time so they can't do other operations work. So, I'm going to have to hire a lot more operations engineers because I have more developers than I have operations. That also can cause you some cost increases, but is it necessarily the way that you need to go.

What we really need is high levels of transparency. You know when we say DevOps means that the ops engineers have to be high-impact members of the project team, they have to be fully engaged in the project, we don't mean that they have to be sitting next to developers every moment of every day in every meeting doing nothing else with their day. We mean they have to have high visibility into the project and know what's going on.

I have seen scenarios, and this is just a small example, where infrastructure staff over the weekend upgrading a network or something like that, made some network change that broke all the builds of my development team. Twenty something builds just failed. And my developers come in on Monday, and they see that all the builds are broken. And they spend three or four hours trying to figure out what caused that, eventually track it down to a network change that hadn't been there on Friday, and then go and contact the infrastructure team and say, "Well, what happened?" and get that remediated. The reason that was able to happen was because of a lack of transparency.

Now, those infrastructure engineers making a network change did not have to be on any of those project teams to know those projects. But if they had been tapped in to our continuous integration server, and they had seen that the minute they made a certain networking change, one minute later twenty project builds failed out of the blue, they wouldn't have to know anything about those projects to know, "Hey, maybe this network change caused this problem." If they had full visibility into all of the operations including the development operations, the DevOps operations, of the company, they could have seen that, engaged their expertise, fixed that problem before it started. And we wouldn't have had that wasteful cycle.

So, we're really just talking about transparency. And I think when you do it right you reduce a lot of that waste and a lot of that waiting for feedback. And you're going to decrease costs.

Hasan Yasar: Right, actually it's a long run DevOps is saving money. There is no real extra cost. It's saving money. It's one of the things for DevOps makes easier for the continuous deployment which we are going to talk about it. So, when you deploy and if you're able to roll back your changes, so think about how much-- if you're not able to roll back your changes, how much money you're going to lose in the business. How much time are you going to lose for the defect and look for what happened? If you have a DevOps model implemented, it's going to save the time and money, which you can go back easily. You can fix easily.

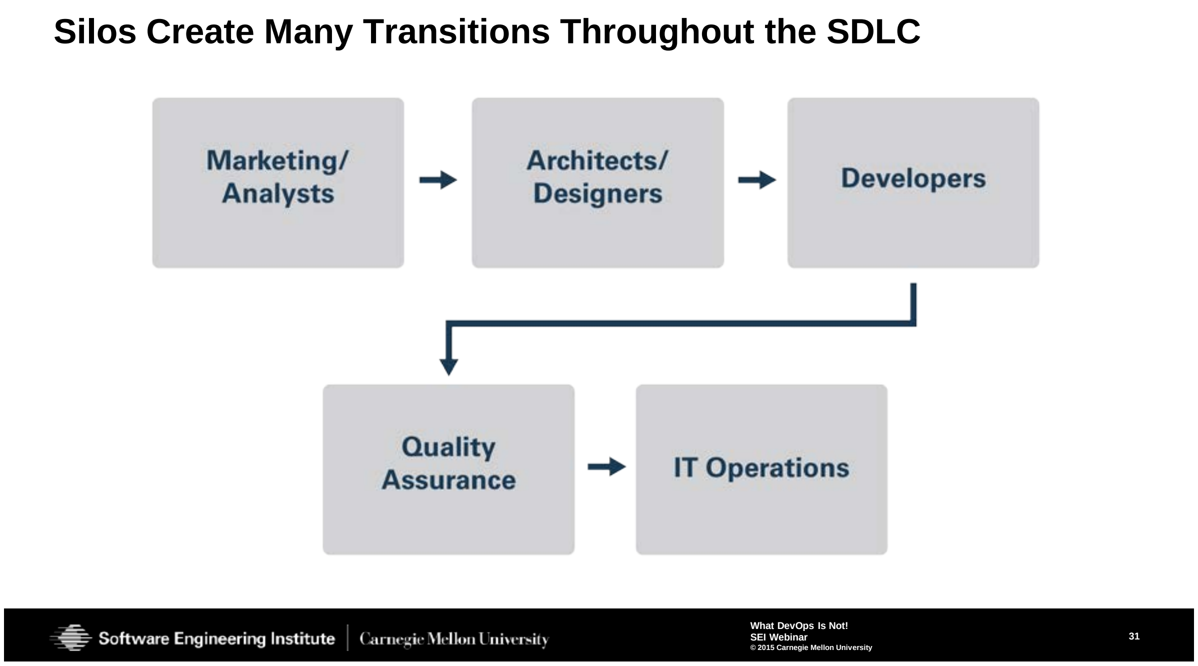
You know what happened. You can get at the machine maker as early stage, then it's saving the money long run. Maybe lump sum ramp up cost happened because it depends on tools maybe depends on some learning curve.

C. Aaron Cois: There's always a learning curve.

Hasan Yasar: It's only a learning curve, but once you start a learning curve as a team together it's going to save the money down the road.

I guess we can go the next-- it's a nice segue for the-- we are talking about almost-- like DevOps and all those things. But it's not the only the operational and development together. It's more than that.

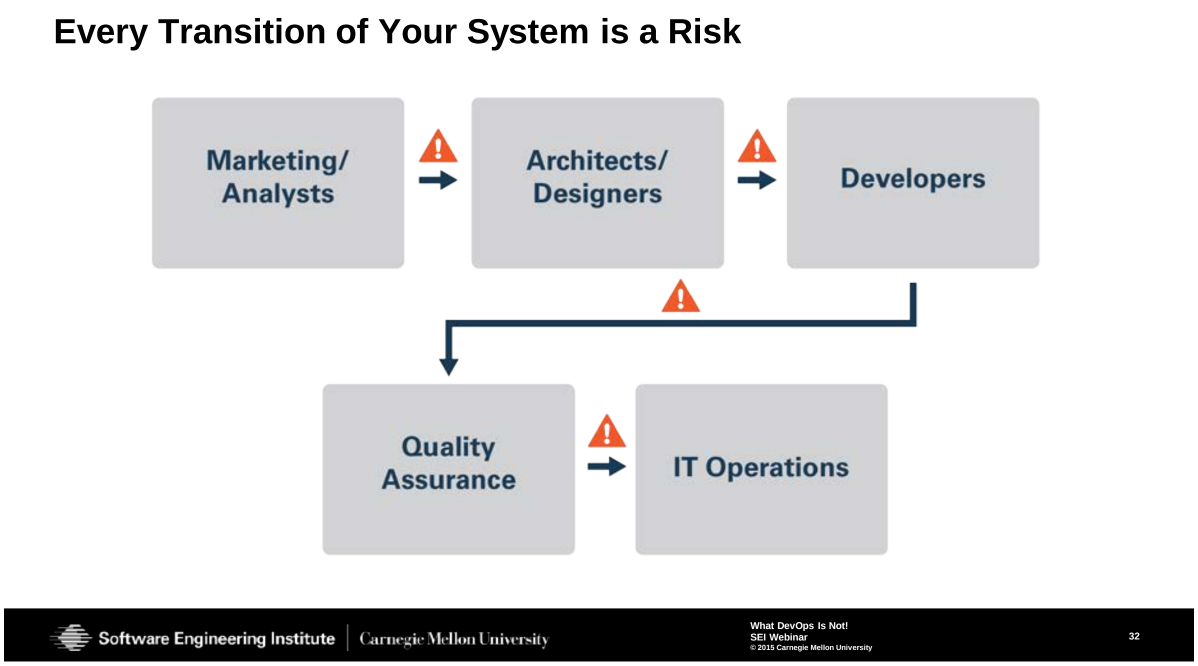
Silos Create Many Transitions Throughout the SDLC



Silos Create Many Transitions Throughout the SDLC

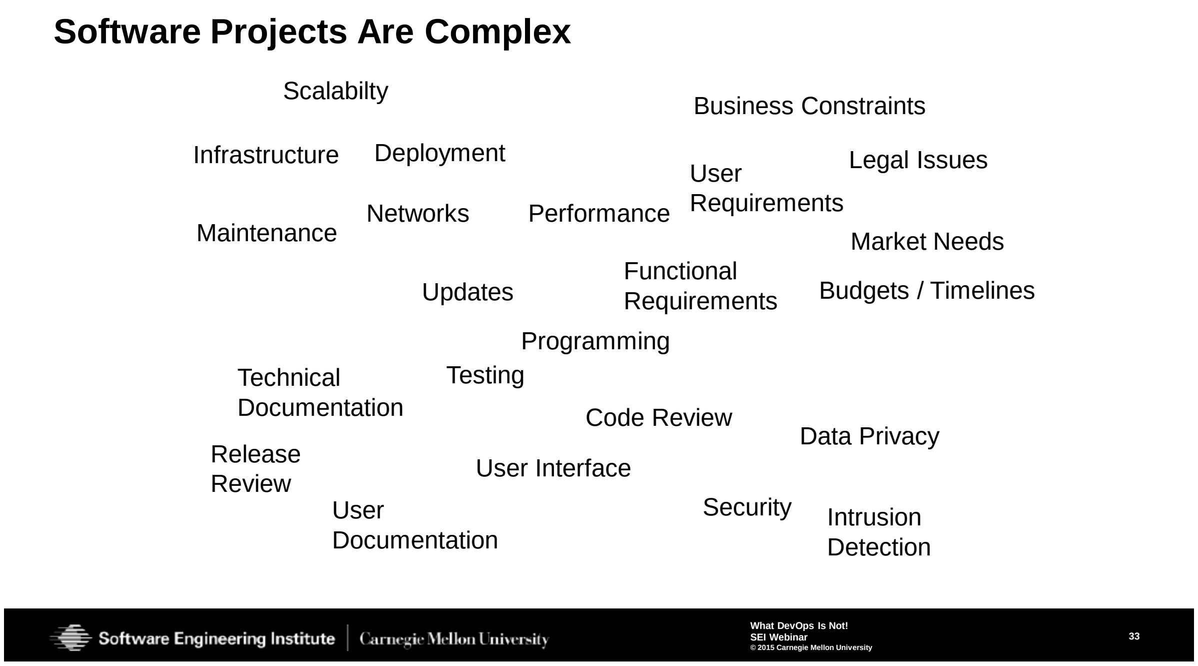
**031 If you look at for the typical software development lifecycles there are a lot of other teams also involved in the SDLC or the team-wise like marketing analysis team, architecture design developers, quality assurance people, and IT operations. So, look at in detail and as group-wise if they're going to share the-- any artifacts between the--

Every Transition of Your System is a Risk


Every Transition of Your System is a Risk
Marketing/ Analysts
Architects/ Designers
Developers
Quality Assurance
IT Operations
Software Engineering Institute
Carnegie Mellon University
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University
32

**032 Group by group or team by team, there's a lot of risk in between, which is creating like any design and which is not addressing an operational needs. Or the marketing thing is creating some user needs. And if they're not able to design very well architectural team, they cannot transfer the whole operation. It's creating a lot of risk in between. So, it's not only getting the operation and developer together. It's more than that.

Software Projects Are Complex



Software Projects Are Complex

Scalabilty

Business Constraints

Infrastructure

Deployment

User Requirements

Legal Issues

Maintenance

Networks

Performance

Market Needs

Updates

Functional Requirements

Budgets / Timelines

Programming

Technical Documentation

Testing

Code Review

Data Privacy

Release Review

User Interface

User Documentation

Security

Intrusion Detection

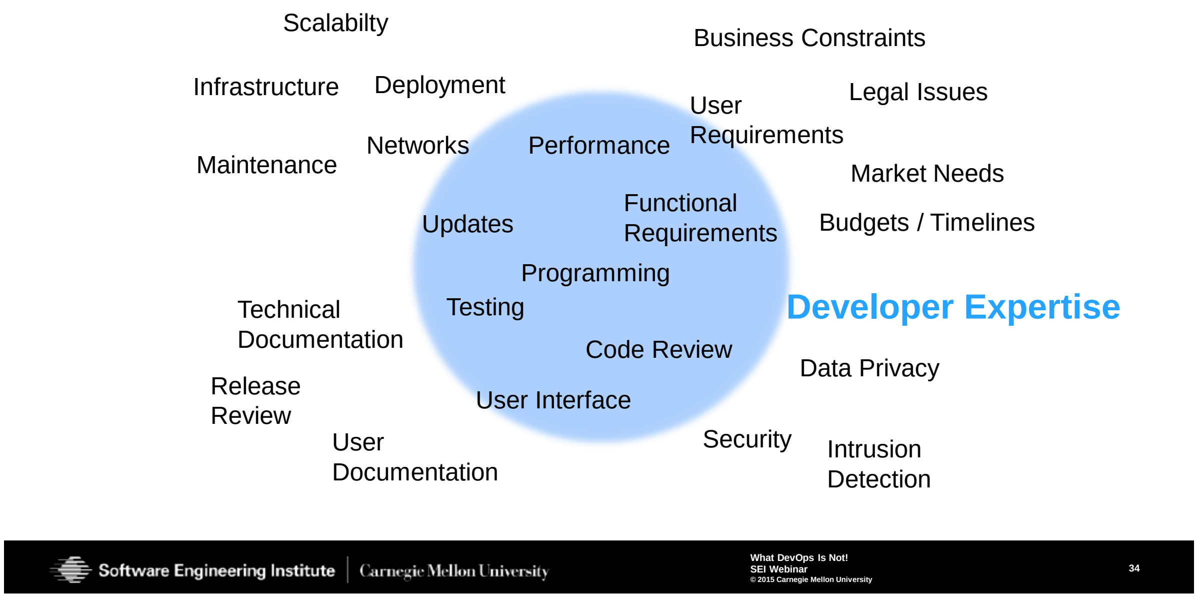
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

33

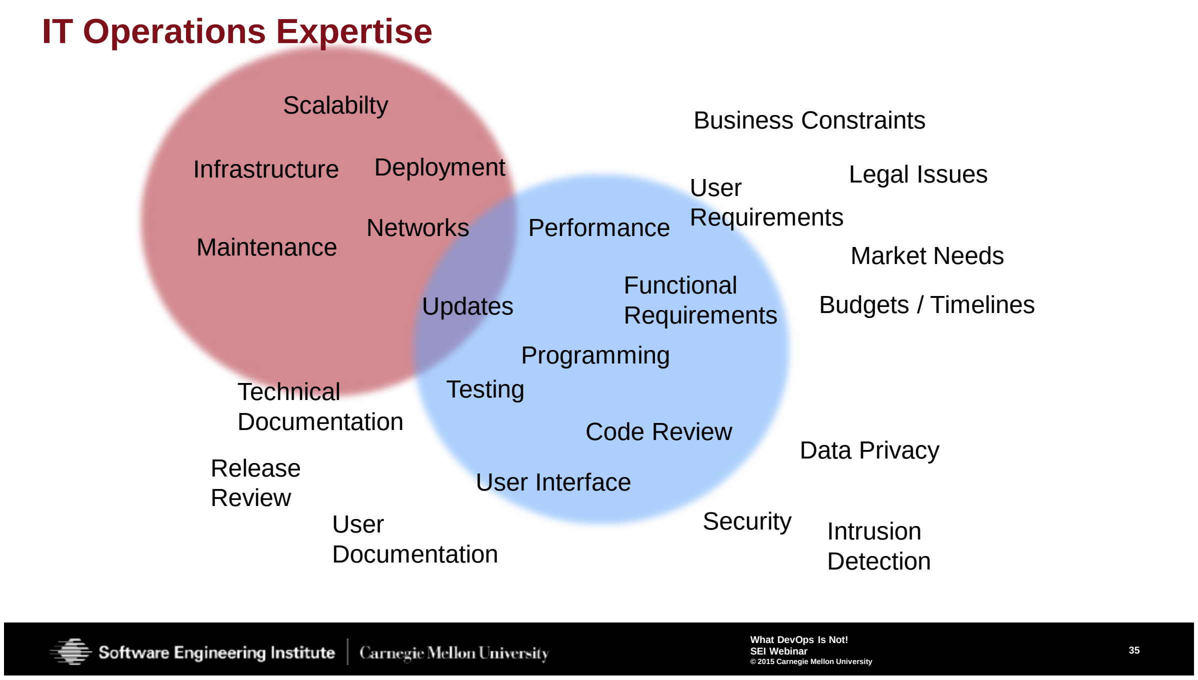
**033 If you look at much closer in the software project, it's really complex thing software project. There are a lot of sub-components into software things like programming, updates, and business constraint, legal issues, marketing needs, data privacy. There are many. We can throw many other quality attributes into the software projects. So, if you going to look at closer--

Scalabilty



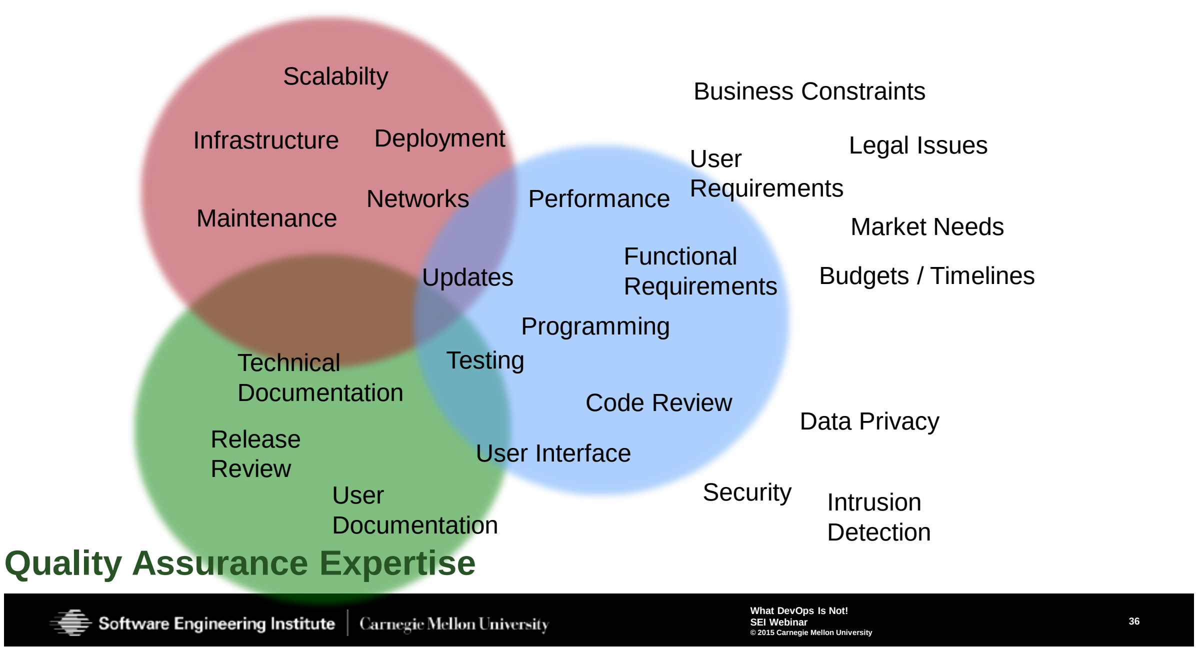
**034 Then we can group them by expertise. It's like developer experts can be combined, some of them, which is the main things are programming.

IT Operations Expertise



**035 That can be IT operation expert is so that can get scalable the infrastructure needs.

Scalabilty



Quality Assurance Expertise

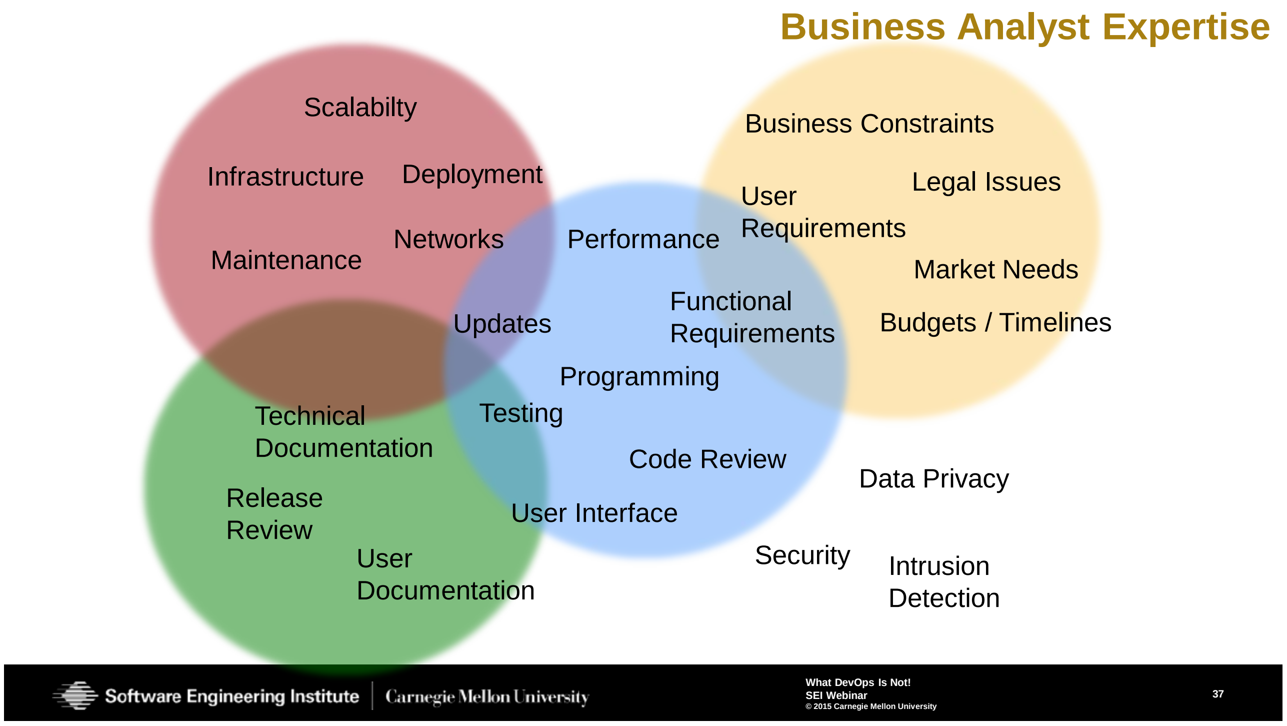
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

36

**036 Or that can be quality assurance expert is so technical documentation, the ways and user manuals and documentations. They are requiring some quality teams together.

Business Analyst Expertise


Business Analyst Expertise

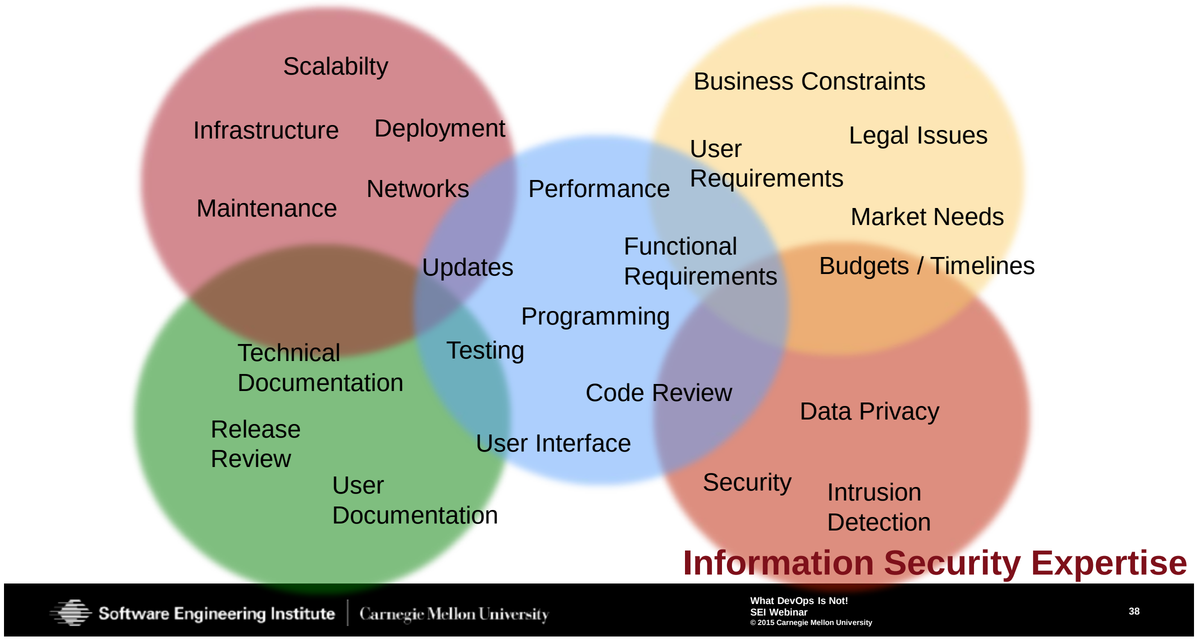
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

37

**037 So, or can be a business analyst expert as which there was one of the question was asking about for the legal issues or the marketing needs, or the some sort of like business constraints. That may require a lot of expertise over there.

Scalabilty

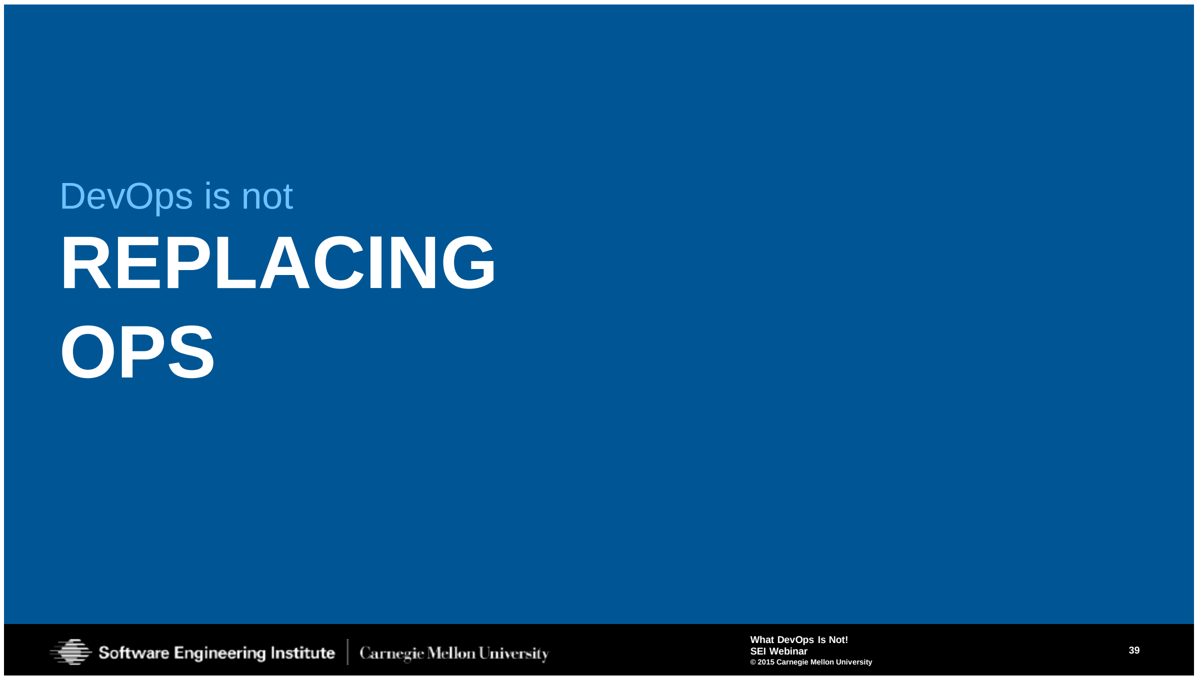


**038 So, if we look at all of the things and maybe for information security expertise, the goal is really getting that work-wise, get expertise into the project. So, when the project is really kicked in, all expertise can say their inputs depends on where they are in the project. So, maybe at the beginning a lot of business constraints, they're going to get their inputs at the beginning. And same thing for scalability, try the development cycles, once they see where are they going, all the project-wise and information security folks can add security at the beginning or during the process instead of waiting at the end to some security analysis or some intrusion detection at the end. So, they can do at the beginning or during the process.

C. Aaron Cois: Because it's too late at the end.

Hasan Yasar: Absolutely, it's going to be too late at the end. It's going to cost a lot of money to at the end. And its goal is real to bring them all this expertise as virtually together. And when they become virtual together, they can give their feedbacks and inputs. So, that's the main purpose of the DevOps thinking and philosophy behind that. So, it's not only the operational team. It's not only the dev. It's all together. It's all together. It's sharing the same goal as business aspect. Sharing the same goals means we have quality software. And it has the business needs, which are addressed properly so we can increase our business values about it.

DevOps is not



DevOps is not

REPLACING OPS

Software Engineering Institute | Carnegie Mellon University

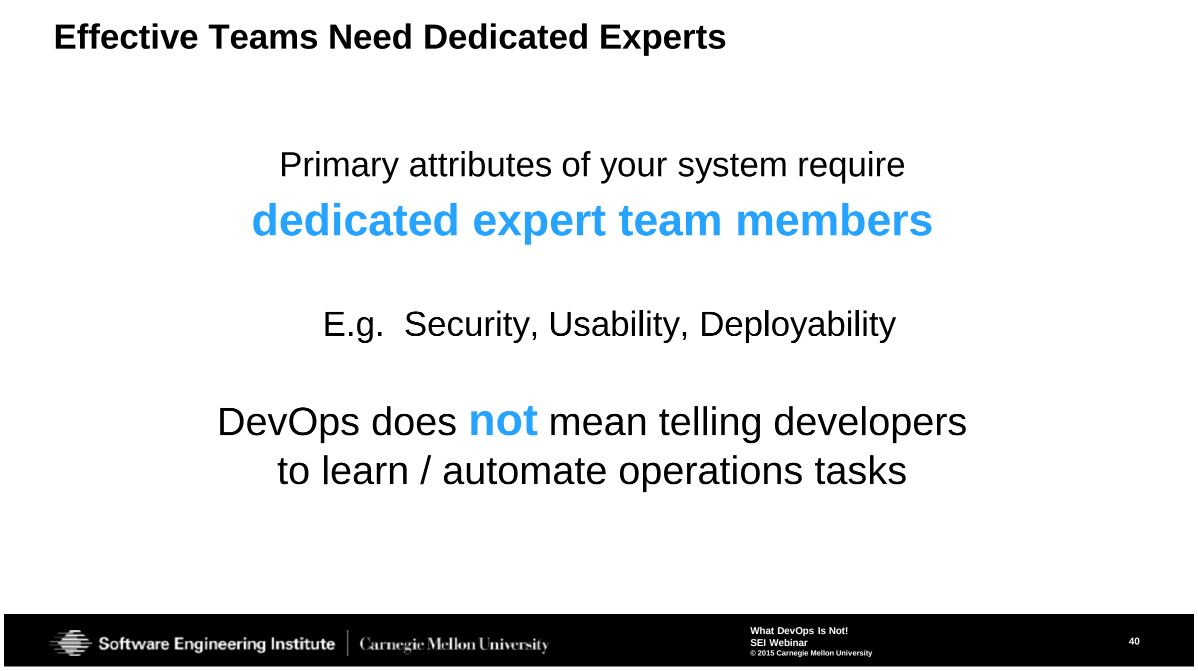
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

39

**039 The other things also we have to touch about we have talked so far, we're not replacing the

operational team. We are getting the
operational team needs.

Effective Teams Need Dedicated Experts



Effective Teams Need Dedicated Experts

Primary attributes of your system require
dedicated expert team members

E.g. Security, Usability, Deployability

DevOps does **not** mean telling developers
to learn / automate operations tasks

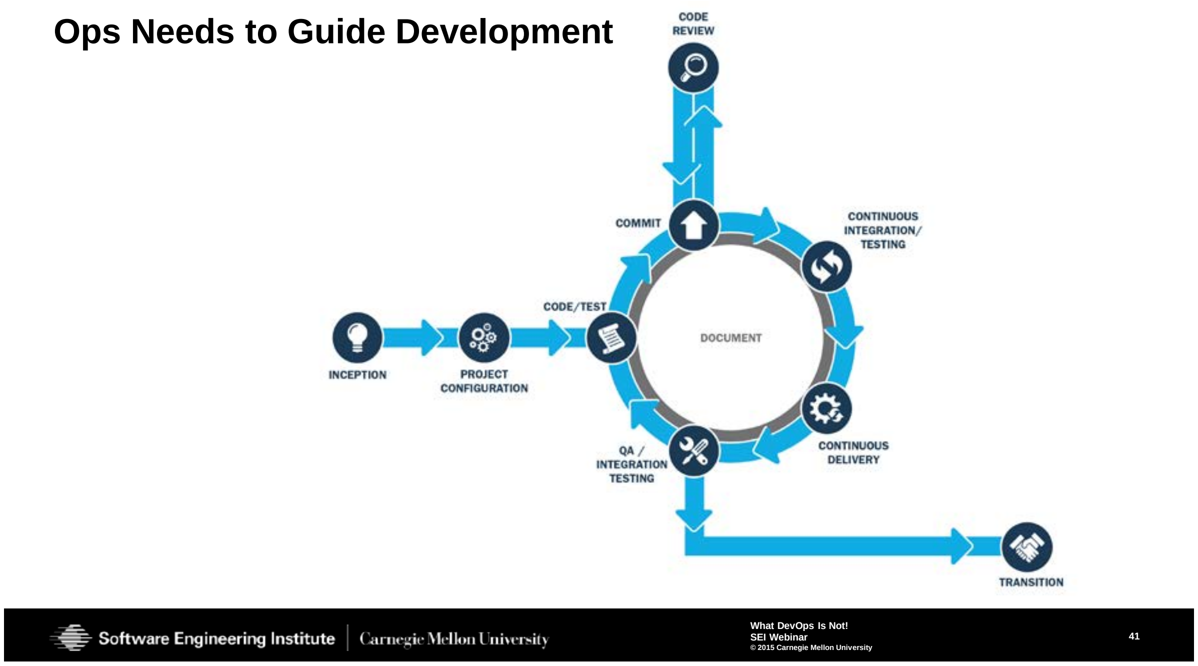
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

40

**040 So, if you think about any
software development process, it is
requiring a lot of decision maker.

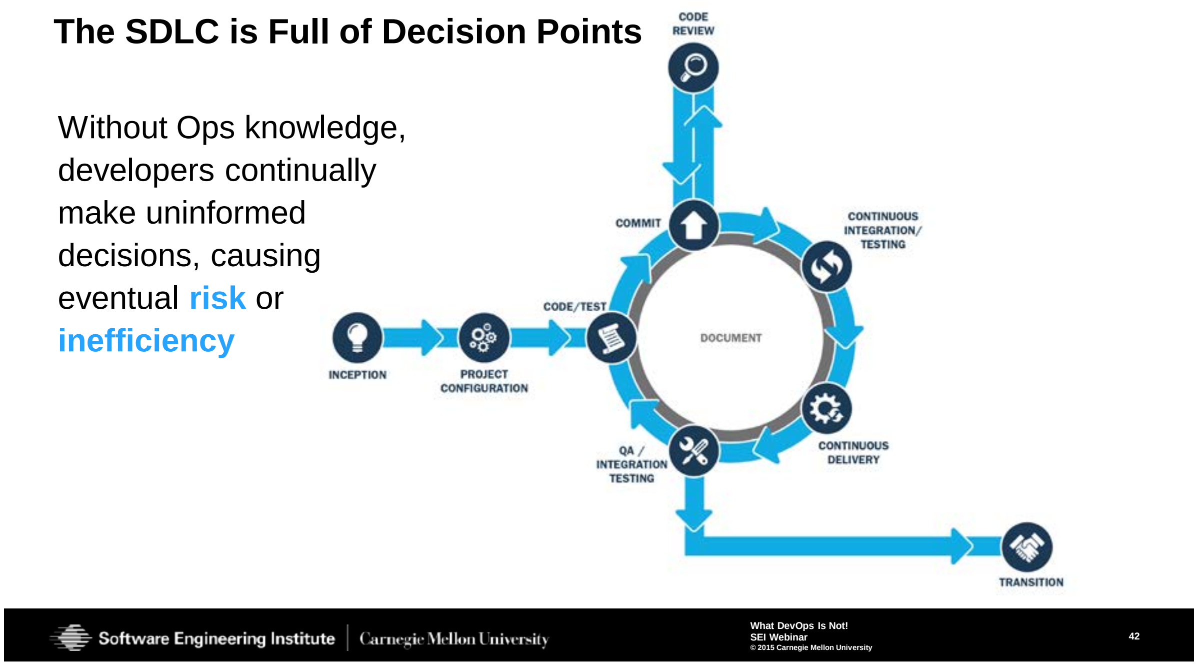
Ops Needs to Guide Development



Ops Needs to Guide Development

**041 And operational team needs
to give their inputs. So, it's a typical
software development process, which
is starting from stage in inception.

The SDLC is Full of Decision Points



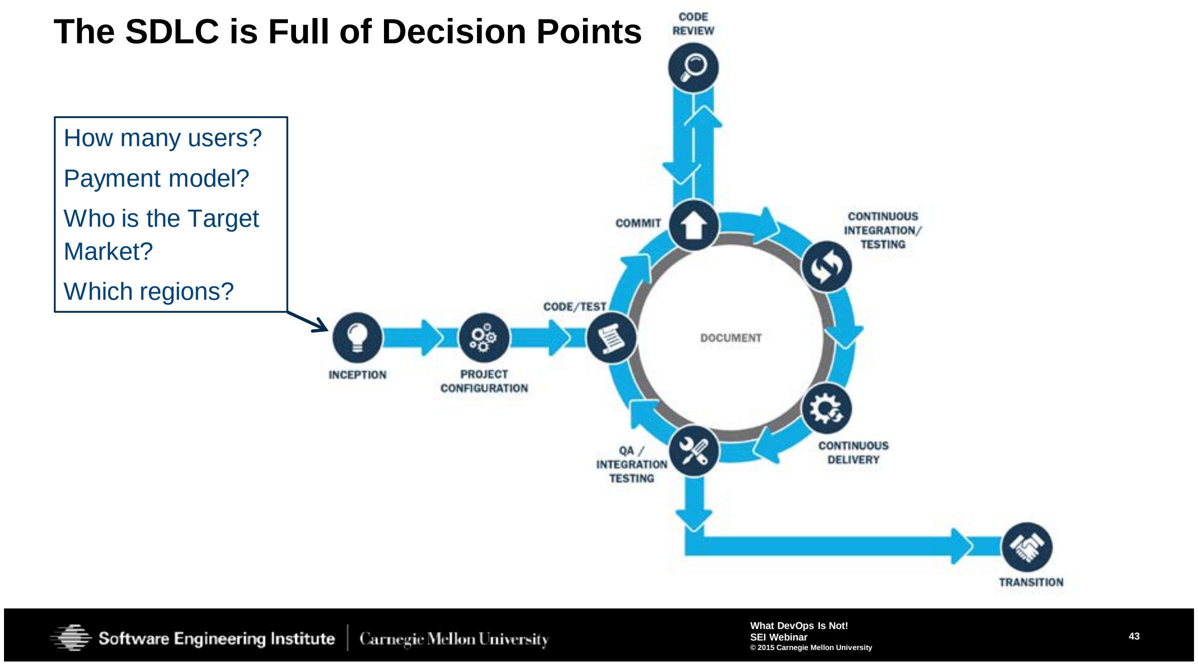
The SDLC is Full of Decision Points

Without Ops knowledge, developers continually make uninformed decisions, causing eventual **risk** or **inefficiency**

CODE REVIEW
COMMIT
CONTINUOUS INTEGRATION/ TESTING
CODE/TEST
DOCUMENT
INCEPTION
PROJECT CONFIGURATION
CONTINUOUS DELIVERY
QA / INTEGRATION TESTING
TRANSITION
Software Engineering Institute | Carnegie Mellon University
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University
42

**042 Which is getting some sort of requirements at the beginning.

The SDLC is Full of Decision Points



The SDLC is Full of Decision Points

How many users?
Payment model?
Who is the Target Market?
Which regions?

CODE REVIEW
COMMIT
CONTINUOUS INTEGRATION/ TESTING
CODE/TEST
DOCUMENT
INCEPTION
PROJECT CONFIGURATION
CONTINUOUS DELIVERY
QA / INTEGRATION TESTING
TRANSITION

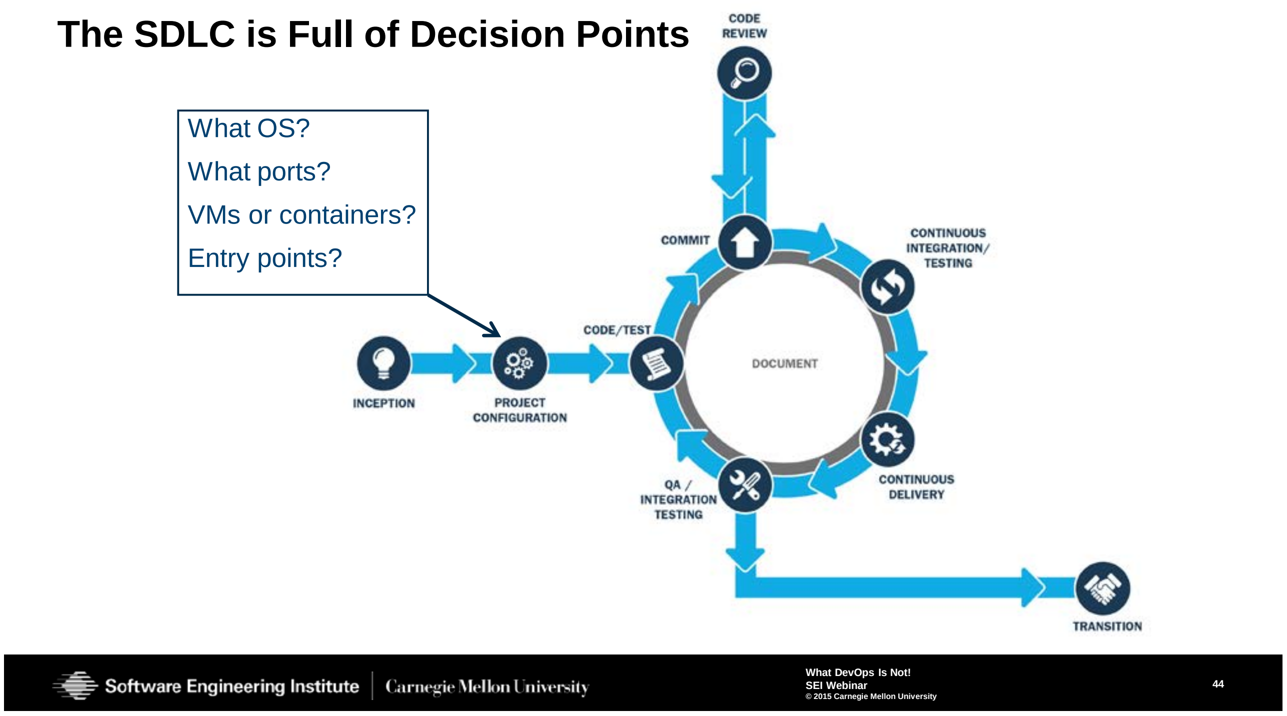
Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

43

**043 And also the project requirements at then. So, these are the very key point. And when we did the requirement scheduling or when we do the some sort of design, operational team, they have to give their inputs so like how many years or what's the payment plateaus and who's going to be targeting the market--

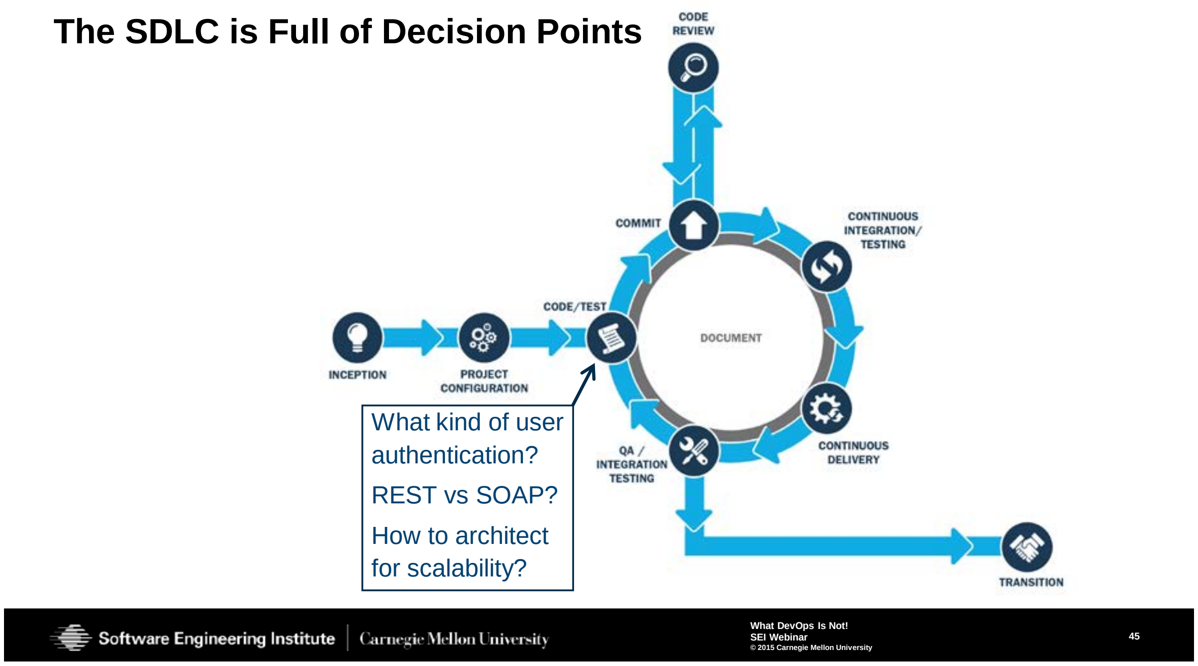
The SDLC is Full of Decision Points


The SDLC is Full of Decision Points

What OS?
What ports?
VMs or containers?
Entry points?

**044 Or other things like what's going to be the OS, what type of ports we have to use. So, if the operational team or the production remains in this team, they approved some sort of OS. And developer will going to use that OS, which is verified. Otherwise, and well typical example is the rule about they're going to use Ubuntu. If the operational team will say no, we're going to use a Santos OS. So, they have to know at the beginning, which is operational team inputs. So, it's really to have operational team inputs at every stage.

The SDLC is Full of Decision Points


The SDLC is Full of Decision Points

What kind of user authentication?

REST vs SOAP?

How to architect for scalability?

**045 It makes much more effective throughout the cycles.

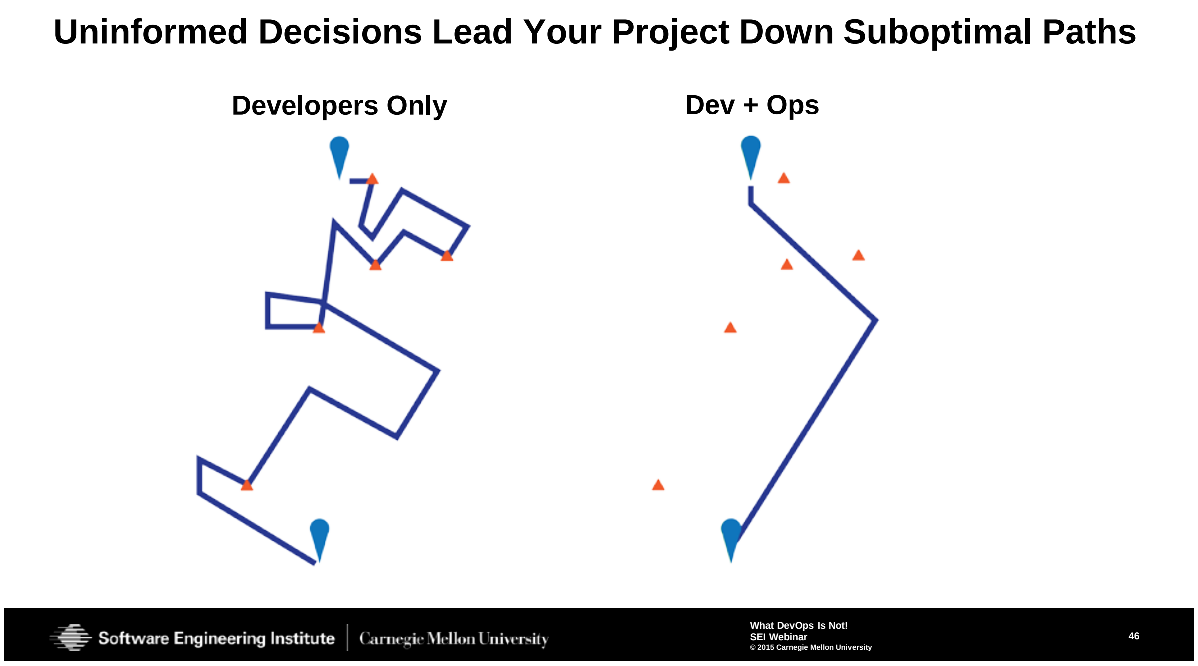
C. Aaron Cois: Right. So, there's a lot of decision points.

Hasan Yasar: Absolutely.

C. Aaron Cois: And if you don't have those operations experts there during those decision points, you'll make a decision that might not meet production needs. And that's exactly what we're trying to avoid.

Hasan Yasar: See, at the end of two things, like if look at all role things, project can be delivered. But project can be delivered in the how much and what's the way?

Uninformed Decisions Lead Your Project Down Suboptimal Paths


Uninformed Decisions Lead Your Project Down Suboptimal Paths

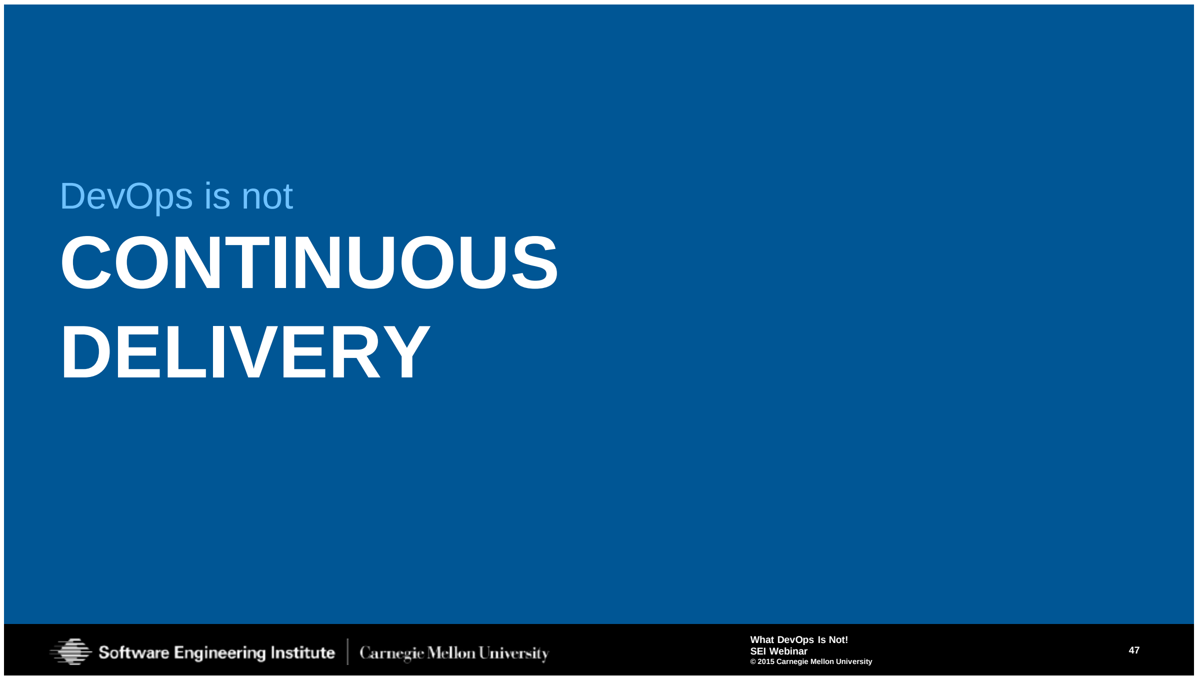
Developers Only
Dev + Ops
Software Engineering Institute | Carnegie Mellon University
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University
46

**046 The goal is getting operational team together. Let them straight path instead of going zig-zag way, and go back and forth, go back and forth, changing. So, let operational team be involved. Get their input at the beginning. Then get their decision maker at the beginning so we can go straight to our project deliverables.

C. Aaron Cois: Right throughout the process, right? Let operations guide development.

Hasan Yasar: Right. It's going to be completely visible throughout the process so that operational team can guide it. DevOps is not taking their jobs. DevOps is literally getting operational team to work together along with other team members throughout the software development process.

DevOps is not


DevOps is not

CONTINUOUS DELIVERY

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

47

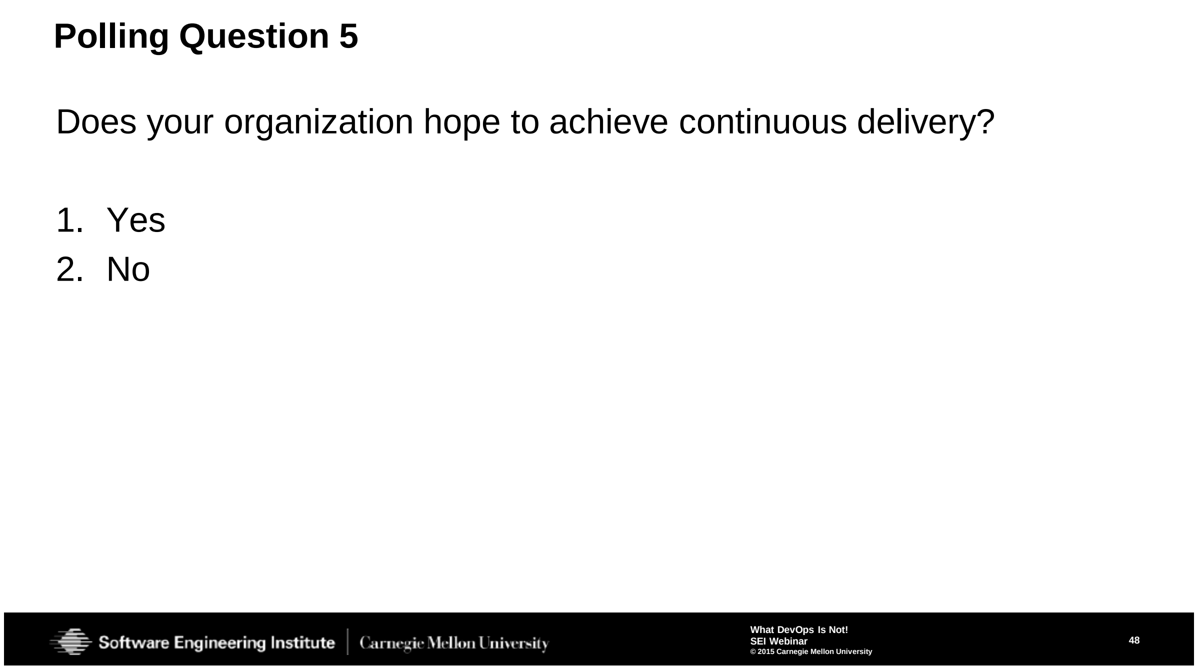
**047 C. Aaron Cois: Right, exactly as you said. Whoever those experts are, those experts all need to be guiding the software development processing. And it's not that we wouldn't necessarily get to the end goal at the end. We might hit the same goal, but we want to get to that goal faster, and cheaper, and more efficiently. So, having that expert guidance is just going to make all that real-time feedback, get those decisions to be decided correctly the first time, and have faster turnaround to get into market.

And this is where you see all the statistics and the improvements in getting features to market faster because you have that guidance and because you have then the levels of automation that allow you to check and make sure that guidance is all followed because you codify it. Then you see turnaround in features to market much, much, much faster because there is not that ambiguity that always exists with well does it actually do what the business needs. And you have to ask everybody in the business.

Hasan Yasar: All the quality attributes during the early process. Operational team can really help them get the security at the beginning, get aware of what should be the design, and addressing all the security nets including SQL injections, some other things as well. So, include in the process instead through waiting at the end of the penetration testing. Let them in at the early stage, which is operational team can give their inputs as early stage, which is what the DevOps is expecting like that.

C. Aaron Cois: Right. We all say it has to be done. You have to address security throughout the entire life cycle. But it's very hard to do. And DevOps gives us ways to do that. And the last things we have, what DevOps is not.

Polling Question 5



Polling Question 5

Does your organization hope to achieve continuous delivery?

1. Yes
2. No

Software Engineering Institute | Carnegie Mellon University
What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University
48

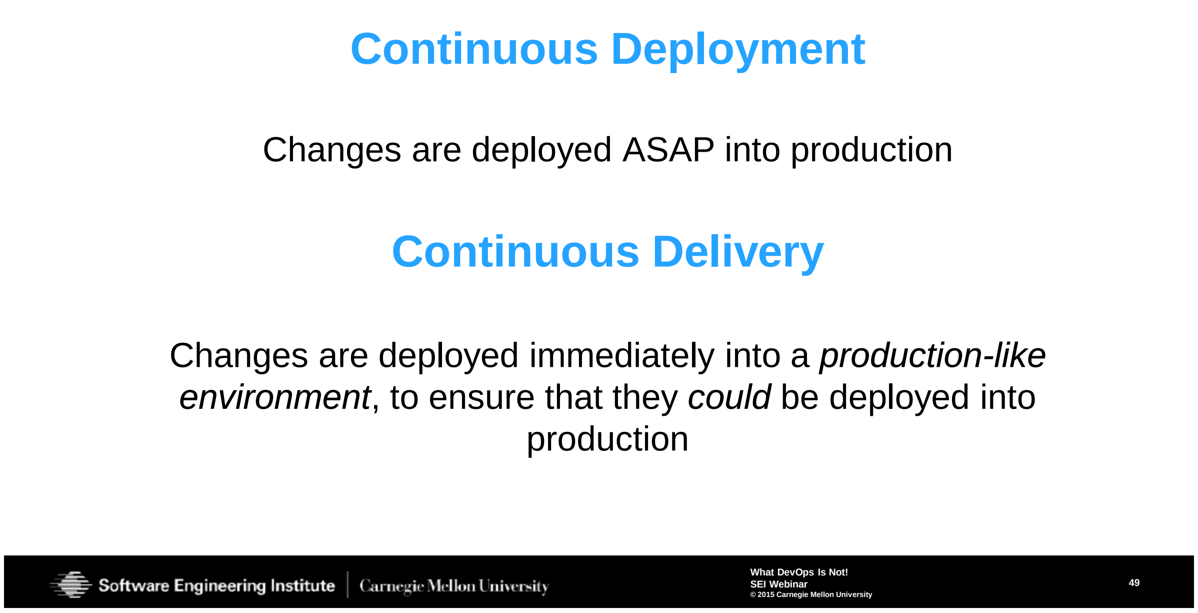
**048 It is not just continuous delivery. We have a poll here as well.

Shane McGraw: Okay. So, that polling question you're going to see pop up now is asking does your organization hope to achieve continuous delivery. And while you're responding to that poll, I just wanted to remind everybody that there was a files tab. We're getting a lot of questions about if this is being archived. And the slides are available from the presentation. You'll see a files tab on your webinar console, which you can download the slides there now.

It is being archived. We'll send that information most likely tomorrow at some point with an email of where to access the archive as well. So, let's look at these results here. And we have a seventy-three percent saying, yes, hope to achieve continuous delivery.

C. Aaron Cois: Okay. So, we have an advanced audience here, which is great to hear.

Continuous Deployment



Continuous Deployment

Changes are deployed ASAP into production

Continuous Delivery

Changes are deployed immediately into a *production-like environment*, to ensure that they *could* be deployed into production

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

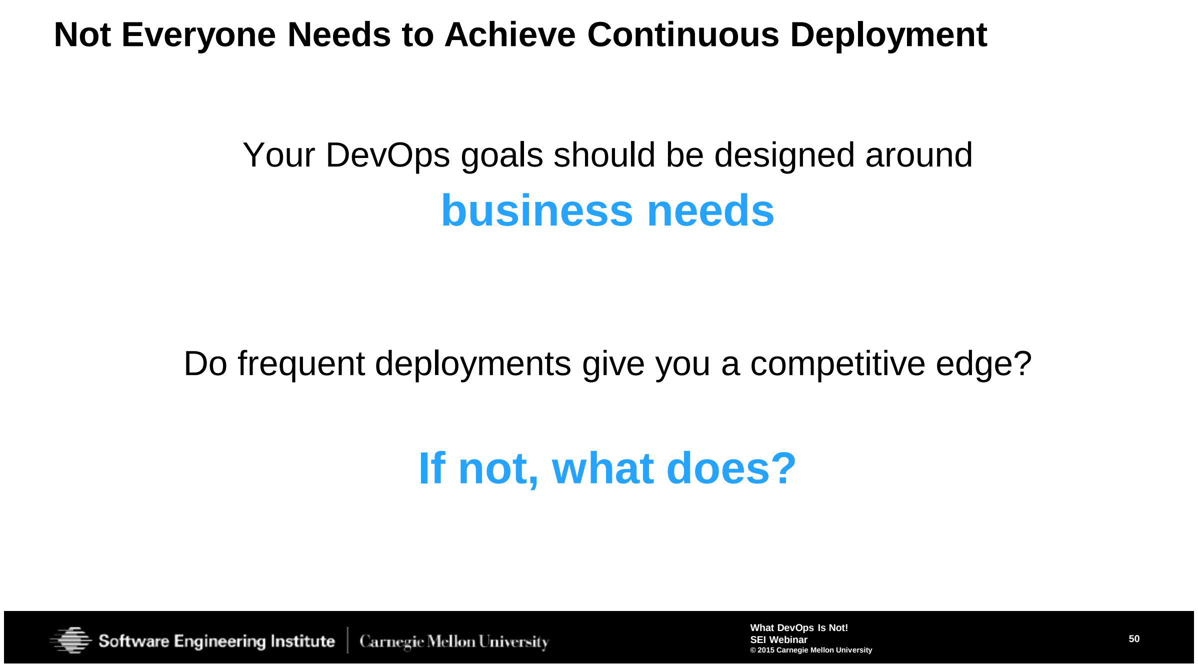
49

**049 First, we just want to go into a few definitions of what continuous delivery and what continuous deployment are because a lot of people, a lot of organizations, I've seen confusion over the definitions. Continuous deployment is what we think of when we think of Amazon or Flickr deploying dozens, hundreds of times a day. That is when changes are deployed automatically into production immediately. Developer develops feature. Feature goes immediately to an automatic pipeline into production. That is continuous deployment.

Continuous delivery is something slightly different. Continuous delivery just deploys automatically all the features into an environment that is production like, a staging environment that has high amounts of parity to production. Continuous delivery is defined by getting the product into a state where we are extremely confident that it could be deployed into actual production, but not necessarily deploying it into actual production.

And these concepts are different because different organizations have different needs. A number of organizations I've talked to--

Not Everyone Needs to Achieve Continuous Deployment



Not Everyone Needs to Achieve Continuous Deployment

Your DevOps goals should be designed around **business needs**

Do frequent deployments give you a competitive edge?

If not, what does?

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

50

**050 Have been concerned because they're a little bit hesitant, a little bit scared about the concept of continuous deployment. And this can be for a number of reasons. One, maybe we don't think we can get there. That's worrisome to have any developer be able to push a change into production. Or maybe, we're in a highly regulated environment. We need to do security checks. We need to do regulatory checks before we push out there. We simply can't do this.

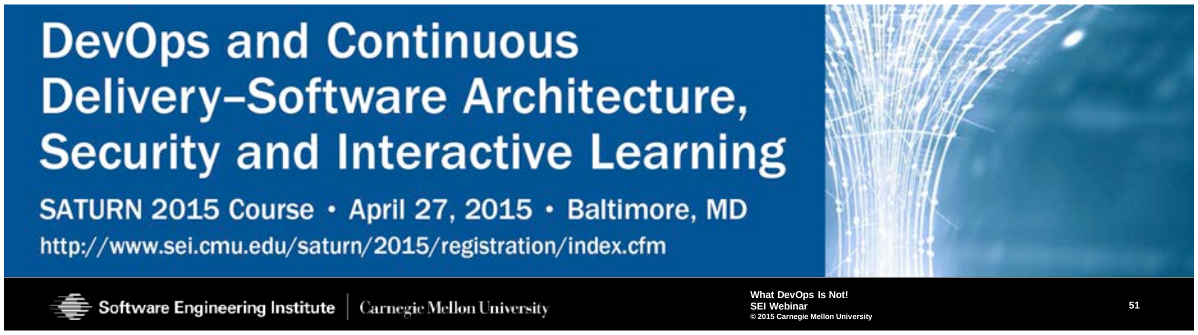
And then the conclusion is we can't do DevOps because we can't do continuous deployment. And that's absolutely not true. DevOps is, at the heart, as we've said a number of times, to be designed specifically to meet your business needs.

So, if frequent deployments give you a competitive edge, excellent. Focus on continuous delivery. Move on to continuous deployment because that's going to provide your business value. But look at your individual business. Look at your environment and say "is that what gives us the competitive edge?" If not, ask the hard question. What would give us a competitive edge in our market? And direct all of your DevOps efforts toward meeting those goals.

It is not always continuous deployment, though it can be a very powerful competitive edge in a number of markets.

Hasan Yasar: We can go to a question and ask if there's any questions.

Q&A



DevOps and Continuous Delivery–Software Architecture, Security and Interactive Learning

SATURN 2015 Course • April 27, 2015 • Baltimore, MD

http://www.sei.cmu.edu/saturn/2015/registration/index.cfm

Software Engineering Institute | Carnegie Mellon University

What DevOps Is Not!
SEI Webinar
© 2015 Carnegie Mellon University

51

**051 Shane McGraw: Okay, well we have got a ton of questions. I think we're going to have to bring you guys in just to do a question and answer session.

Hasan Yasar: That'd be great.

Shane McGraw: We'll get into that now. So, let's dive right in. one from John asking in a larger organization, what level of management sponsorship is needed to successfully break down silos and implement DevOps.

C. Aaron Cois: That's a hard question because in a lot of cases, my ideal answer is complete. DevOps works best when there is complete organizational buy in because ultimately, you need everybody in the organization to feel free to communicate, to collaborate, to share information, and not to worry about whether that's going to have any repercussions or ramifications. And that needs to be, not only supported, but sometimes enforced to build that culture. That is one of the trickiest parts.

Hasan Yasar: That's actually requiring some top and bottom approach. So, the bottom up people like developers, they have to have some sort of understanding the flows of the data what they're going on. And management team, they can measure it. If the management team knows how much money they are saving in having a DevOps, they going to jump on the board immediately, which as C-level people, they can see what's really going on. They can get their buy in. and developers arranged in the operational team, other team members, they know what is really DevOps means. So it's kind of like all together, bottom up, top down. So, you're going to get in the middle in central place that can share and continue like that.

C. Aaron Cois: The other thing you see is we said you need that expertise in business needs. You need to know exactly what metrics and things are going to provide your business value. Well, who knows that? That's usually the upper level management, the executives. So, they need to be a part of this. They're stakeholders. And they're experts. That expertise has to be

worked in or you're not going to hit
your targets.

Shane McGraw: Okay, great. From
Steve asking how can we shift the
thinking/culture of an operations
team who is reluctant to put new
releases into production because
their primary goal is maintaining high
up time.

Hasan Yasar: I can address
actually. The operational team, if
they have a really complete
understanding of how DevOps is
going to help for them, they're going
to say, "Okay, I should get it right
away," because that's their primary
goal is keep them up time because
when they are getting some
productions up and running. But if
their product, the operational team
gets as completely test it, which
they're all aware of it, they know
already at the beginning, they know
what is coming the day it shoves.
They know in and out everything. So,
developers are all engaged with
operations, instead of handing wash
and say that's not my responsibility.
It's your responsibility, which is a
problem here. So, the operational
team will get more helpful, more
benefits having DevOps like that.

C. Aaron Cois: Right, it's about that
consistency. My answer would be
that operational team, what it sounds
like to me is that high availability,
that that up time is a very important
quality attribute to your organization,
assuming that's an organizational directive.
Then that's important to know.

So, the way I would go about that is to start developing DevOps processes, automated deployments, automated systems to go into a staging environment. And when operations gets comfortable seeing that that staging environment has really high up time, that it doesn't take us a long time to push that deployment, that that deployment--these deployments are not failing, they're not causing any--

Hasan Yasar: Or rollback this and other things.

C. Aaron Cois: Right. Right. Rollback, automated rollback, see that when we do a deployment, if there was a failure, it automatically rolls back to the last version. You see no hiccups in up time. When they see that, they will adopt it. But it is a valid concern to say up time is very important for us. And these deployments, every time you go for a deployment, it takes the system offline for twenty minutes. No, they're not going to want to do that multiple times a day. But I think it's all about proving the capability. And then that capability can roll into actual production. Delivery before deployment.

Shane McGraw: Great. Just before we get the next question, just a quick reminder of everybody to click on that survey tab before exiting as your feedback is always greatly appreciated in how we can improve our events here at the SEI.

Question from Ray asking any example of challenging managers to work across lines, i.e. outside their own organization. Once again, any examples of challenging managers to work across lines, for example, outside their own organization?

Hasan Yasar: It's a contracting shop basically. It's a good example. The contracting, which one organization has contracted another organization maybe for a couple of reasons. They can have their own developments. They may be outsourcing development modules to somebody else, which his another challenge over there. So, DevOps is going to bring that again the same philosophy of share together using the collaboration portals. And if the management team knows to monitor it, all the process, all the development process. It's going to be much easier for them to control it. That's management buy in or whether I can add it.

C. Aaron Cois: Sometimes it comes down to what-- how people are-- how they're measured, how they're monitored, what are the incentives in your business. So, how are those managers' performance measured? And what does that mean for your organization? And sometimes, you have to look at that. If they aren't incentivized to work across, it may be because they're managed in different ways. And that has to be addressed and looked at to give them those shared goals. Managers need to share goals just as much as the technical folks.

Shane McGraw: Okay, folks I know we're at 2:30. We're going to take one more question. I just wanted to remind everybody before we go on to that last question to make sure you look into the files tab. You're going to see a lot of great resources in there. The CERT DevOps team here recently started a blog series for the last, I think, five or six weeks, a bunch of great posts. You'll see a file there that will take you to those blog posts.

Aaron is co-instructing a course. It's Saturn 2015 on continuous integration in DevOps. So, if you're in the Baltimore or D.C. area, and DevOps and continuous integration is an area of interest, be sure to look at that course. But these are all materials you can walk away with today.

So, last question we'll wrap up with is from Dan asking does a DevOps group make sense for a software vendor with many software products as a means of enforcing a common DevOps approach across product teams in separate locations.

C. Aaron Cois: So, that's interesting. So, does a DevOps team, and independent team you're saying, make sense when you have a lot of different products? We operate in a world like that. We have a lot of different projects going on simultaneously. And we have a pretty robust DevOps solution because of that, to keep things in line. So, the idea of keeping things in line makes a lot of sense.

But the question is what are the roles within the projects of that DevOps team. Are they infrastructure engineers? Are they developers? What are they doing and how are they engaging? Or are you just talking about someone who oversees process and advises on process?

I think what Hasan had said before. This concept of a virtual team, the concept of matrixing makes a lot of sense in that context. If you get everybody to have buy in. say, all the infrastructure people get together in a matrix team and decide on how the DevOps workflow should work. That can make some sense.

Hasan Yasar: Actually, I think a couple other things as well here. Based on the current research in 2016, it will be a DevOps year. Research says that. One of the things that's confusing for DevOps is not to be discussed that was in the product. But there are a lot of things which is DevOps enabled tools, DevOps enabled process, DevOps enabled project. We have to think that way, DevOps enabled so that means you can do something with DevOps. And it makes you to help, not really as the group, not as a team, not as a product, it's enabled. So, that means it's a process that might work because my project is going to be compatible with what I'm thinking. So, can I achieve the goals? That's a key point. We have to think about that.

C. Aaron Cois: One thing I will advise is that most processes, specifically DevOps processes, do work a lot better when there is input from the front lines on that. If you just externally create a DevOps process and then just try to place it down and force it across all of your developers, all of your infrastructure people, you will get kick back. And you will not get full engagement and buy in in that from below. And you're going to need that as well.

The project teams are the experts in what their projects need. So, getting them to be involved in designing the DevOps process is going to yield benefits, not only because the process could better, but because they will be engaged in and have ownership in it. So, they will then enforce it themselves.

Shane McGraw: Okay, folks, that is going to wrap up our presentation for today. Again, thank you for attending. We apologize for the technical difficulties at the beginning. And again, this will be archived for anybody that missed anything at the beginning. The link will be sent out tomorrow. And we apologize also for any questions we didn't get to. There's lots of great questions left in the queue. So, we'll make sure we get them to Aaron and Hasan and their team to review and hopefully get a response out to you that way.

Thanks, everyone. Have a great day.