Agile and DevOps: Your Questions. Our Answers

Table of Contents

Agile and DevOps

Agile and DevOps:
Your Questions. Our Answers

Eileen Wrubel

Hasan Yasar

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Carnegie Mellon University

Carnegie Mellon University

Copyright 2018

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-0298

Agile and DevOps: Your Questions. Our Answers

**001 Presenter: And hello from the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania. We welcome you to Virtual SEI. Our

presentation today is Agile and DevOps, your Questions our answers. My name is Shane McGraw, your moderator for the presentation. And I'd like to introduce our two panelists for this afternoon. First, we have Eileen Wrubel. And Eileen is the Technical Lead for the Agile in Governments program here at the SEI. Eileen, Welcome.

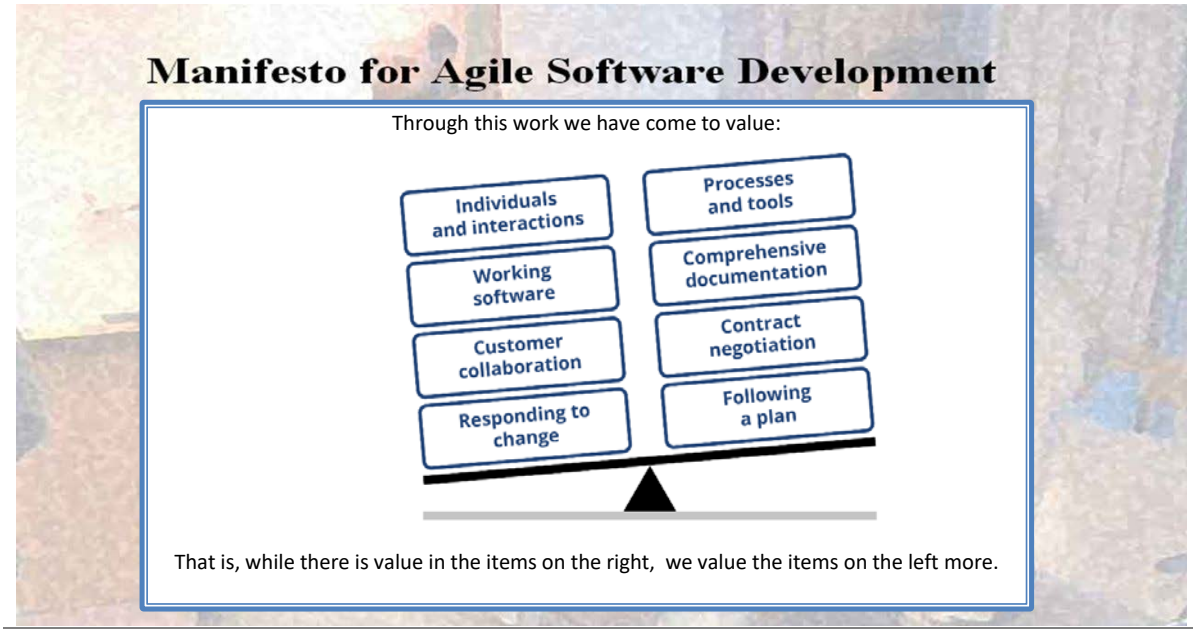
Presenter: Thanks, Shane.

Presenter: And also, our next panelist is Hasan Yasar. And Hasan is the Technical Manager for the Secure Lifecycles division here-- within the CERT division here at the SEI. Hasan, welcome.

Hasan: Thank you, Shane.
Presenter: So, we have lots of questions coming in folks. So, we're going to not waste your time. And we're going to dive right in to it. We've got about thirty minutes. So, we will get as many questions as we can. With Agile and DevOps getting more popular in the Department of Defense in federal government space, what is the SEI's take on these terms? So, if we can lay a little groundwork before we get into audience questions. So, Eileen?

Presenter: Sure. So--

Manifesto for Agile Software Development

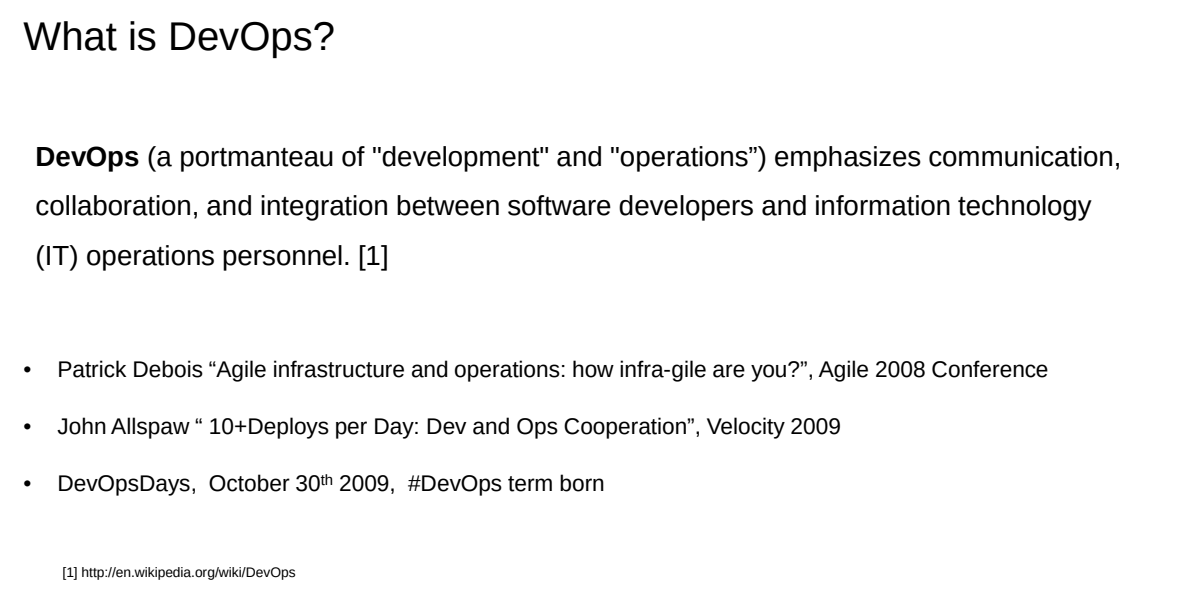


**006 When we're talking about Agile software development, that's something that's pretty commonly understood in the industry to mean that we are talking about small collaborative teams working in small batches focusing on flexibility and responsiveness to change and not building big infrastructure to manage projects, not relying on projective representations of systems but rather relying on working software. Our lens here at the SEI is really helping defense and government organizations change how they need to do business in order to be flexible, and responsive, and work with engineering organizations that are employing these methods because it's not a one to one shift with how we typically have acquired software-reliant systems in the past.

Presenter: Hasan?

Hasan: So, DevOps is-- it's actually the portmanteau for Dev and Ops together. And it means that--

What is DevOps?

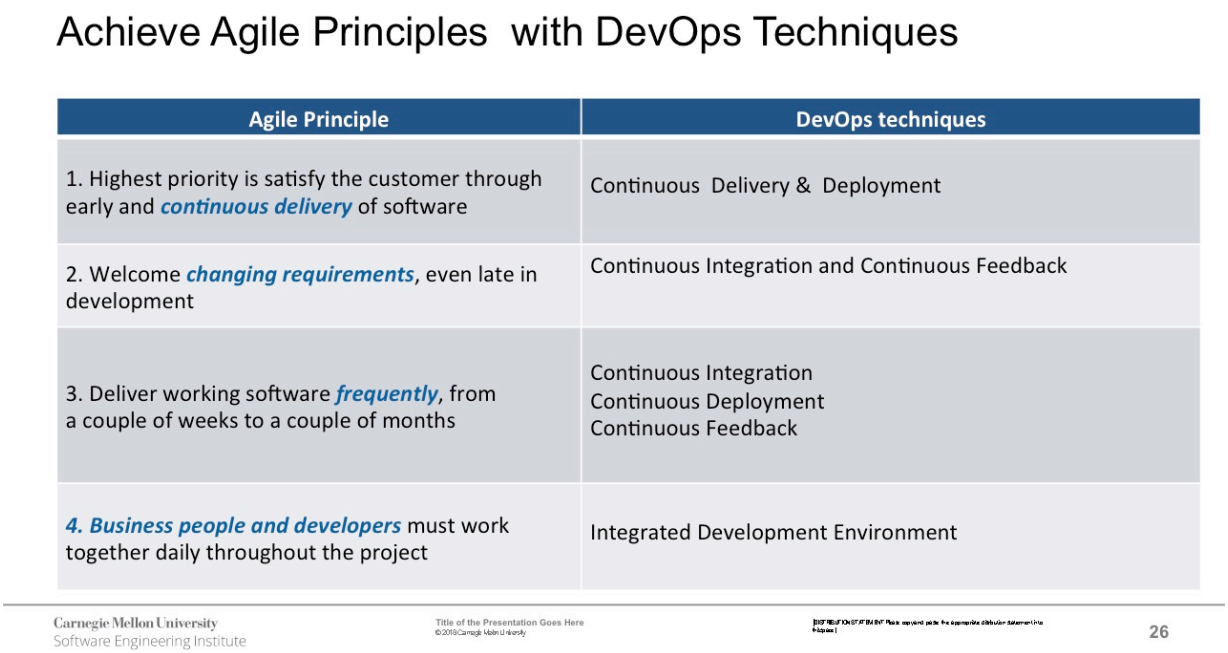


**013 Emphasize communication, collaboration between all stakeholders, not only for dev and ops. And SEI's perspective, especially for DoD space, or the federal government, it is a full communication from beginning to end means like including all stakeholders. It is not only for dev. It is not only for ops. There's acquisition pieces in place, which is not typically shown in the industry practitioner things because industry is not really hiring up and acquiring software things. But DoD is acquiring the software capabilities. So, taking that approach and what the DoD folks need to know upfront and to make DevOps compatible, what are

the principles are DevOps, like a communication, collaboration, infrastructure as code pieces are the fundamental key point, include that concept from the beginning. And beginning is like during the acquisition capabilities, or writing requirements, or maybe writing a system design that that can at the beginning, including the testing cases as well, carry out that concept all the way to the production environment, which is like end users. So, that's the things as overall DevOps concept and bring the DoD space.

Presenter: So, we requested questions before the event. So, we have some questions there. If you have a question during the live event here, you'll see a chat tab on whatever platform you're watching on. Whether you're on Virtual SEI or our YouTube page, feel free to use the chat.

Achieve Agile Principles with DevOps Techniques



Achieve Agile Principles with DevOps Techniques

Agile Principle	DevOps techniques
1. Highest priority is satisfy the customer through early and ***continuous delivery*** of software	Continuous Delivery & Deployment
2. Welcome ***changing requirements***, even late in development	Continuous Integration and Continuous Feedback
3. Deliver working software ***frequently***, from a couple of weeks to a couple of months	Continuous Integration Continuous Deployment Continuous Feedback
4. Business people and developers must work together daily throughout the project	Integrated Development Environment

**026 But a first question came in from Connie asking, "How's DevOps different from the Agile framework? What is the relationship between Agile and DevOps?"

Presenter: So, you can think of Agile software engineering as being about the activities where we do discovery and we write code in small batches, and we successively approximate the functionality towards some end goal. We also layer in the fact that that isn't just an engineering problem, but it involves the business infrastructure, how you do contracting, how you work with end user organizations, how you work with testers. To layer DevOps into that, I'm going to let Hasan take that part of the question.

Hasan: So, what I'm taking here is that DevOps is basically an extension

of Agile thinking. If you look at Agile principles, there is the twelve principles. It says how you do the early deliveries and the quick sprint or the iterative release. But it is requiring some sort of mechanics and some techniques because none of the Agile principles says how to do it. DevOps comes in the picture to make that happen. Like give an example, if you look at the first principles of Agile, the highest priority is satisfy the customer throughout early and continuous delivery of software and requiring a continuous delivery or continuous deployment. So, DevOps is bringing that mindset and how we get quickly integrated into the code and release that code pieces to the user, so the user can see immediately as a result of the integration phases. Either delivery can be done in the staging environment, or deployment can be done in a production environment. So, that's basically thinking and concept-wise taking the Agile concept from a development perspective and carry out to the operational side of it because Agile is not looking for specific operation perspective. DevOps is covering up operational perspective all the way to end users and having a feedback back to the Agile team. That's the relationship.

I can describe that. So, maybe we can talk about more, which one is better to other? There is no such which one is better to other. It's complementing each other. So, if you are doing Agile process, you have to have a DevOps mindset, DevOps

techniques to that. If you are doing DevOps, you should have some architectural, some work-wise to make it more iterative release, which is requiring some Agile techniques as well. So, kind of a hand and hand together, it's complementing each other. One is the technical side of it I can describe. One is more about the process, more about the software development activities.

Presenter: So, just a quick comment from Joe asking, "How do you get started with DevOps and Agile? Are there frameworks you suggest?"

Hasan: How you start it, it is-- there is three components actually how you start it. It is requiring people. The people should know about Agile process and techniques. And how you do the Agile scrum, maybe Kanban, whatever the practices, they are following Agile principle, which I'm going to hand it to Eileen who can give more. But for a DevOps perspective, it is requiring a platform because it's running in some tool set, and also changing the mindset, how you communicate, how you collaborate.

So, basically DevOps has the four fundamental principles like the communication, collaboration, infrastructure as a code, automation, and monitoring. So, to establish the four principles, you're going to start from the first one is communication. Look for the team and how are we going to communicate to each other.

Like specifically, if you are getting requirements, or if you are looking for any features that we are going to develop, how we are getting that features from a team, maybe user directly, then how we are carry out that feature throughout the lifecycle. Like how we are communicating with the dev team. When we talk about the dev team, do we talking with the operational team as well in terms of requirements? Means like what type of system requirements that my code is going to run in terms of security, in terms of reliability and scalability. Who's going to tell me that here's our requirement because usually the user is not going to tell you the requirements? They will tell you the function of the requirements, which is another way of communication with the rest of the team members. So, what I suggest, start how we improve your communication channels to make sure you're on the same page.

Then build up the right tool stack. It's another important thing because we cannot do the automations without having a supported platform. Then the platform should support that automation including testing, including continuous integration, continuous delivery, continuous deployment, which as a tool selection comes into the pictures. Then also the process is related because DevOps is not being a code by coding. It is more about a discipline approach. It doesn't mean that you go to a website and pull the open source, and your stuff,

and make that happen. No, it is requiring some discipline. Discipline means one basic thing. If you are doing the code checking, or code repository things, you've got to check in daily for having continuous integration kicked in at the end of the day, so you have a continuous build process, which is one basic principles. So, Agile things, maybe I can throw more stuff, Eileen.

Presenter: I think you covered a lot of it. As far as frameworks for starting with the processes, the things I would suggest are-- there's a lot out there already about team-level, Scrum, and Kanban kinds of practices, extreme programming, test-driven development. There's also a lot of information out there about Agile for larger organizations and larger projects when you're getting past the maybe two or three teams of engineers. My colleague Will Hayes and I did a podcast recently on considerations for scaling Agile into larger ecosystems like the DoD. And I'm sure we could make that link available at the end of the broadcast as well. But scaling Agile into the large is a-- there's a lot of approaches to that out there that you can do, as well.

Presenter: So, that kind of goes into our next question. But it's from Ruth asking, "How does Agile and DevOps embrace the integration of large projects into particular projects involving hardware?"

Hasan: So, going back to the first question, Shane, that you asked, it's how you start it. So, once you-- you've got to start from the basic-- like a very small sized project first because it's requiring cultural changes first. So, jumping into the big project is-- you have some preliminary work done first and make that policy changes, or procedural changes in how we get and DevOps into the work and working a small project first. So, once you do small projects, then you can expand that thinking in terms of platform readiness and people communications and process to the bigger scale.

Let's say we had one continuous pipeline, software development pipeline, for one module of that big picture. The big picture is like big software, large systems. Then we can add a multiple pipeline that working for specific groups of that modules as part of the large-scale projects. So, that multiple development pipeline can run parallel, and probably software can see the picture like a scaled-Agile framework can be another process support. So, into DevOps perspective, you may have a multiple pipeline is running for the continuous integration, continuous build process, so you can merge that pipeline into the bigger part, like pipeline of pipeline. Then you can scale up the big projects. So, big project doesn't mean everybody has to be running same and wide, I mean, all the time. But they can stand up their own DevOps pipeline.

And some organization does centralized repositories. Like Google has a perfect example. They have one single repository. And everybody's committing over there. But they may have a multiple pipeline is running it. So, people are able to stand up their environment. And go back to the Amazon cases, they have size team. They're able to scale up huge websites plus including AWS services setting up a small team. But the small team is responsible for design, and develop, and deploy it. So, again, this is kind of like an organizational culture is also important. So, start from the small projects, and carry out the larger projects.

And second portion of the question you were asking about the hardware integration. And in this day and age, typically software is eating up the world. So, we look at the hardware. Still there's software pieces in it. So, taking that hardware concept is a simulated platform like basically having a maxim environment or kind of virtualized platform. In this technology, it is available. So, it may be-- let's say if you're developing a specific application for Android or the iOS. And almost all dev environment has an emulated version of Android. So, you can really test it out. Or if you're developing a software for IoT devices, and IoT has emulator things we can do. If there is no emulators for a specific device, hardware engineers they can work with the software engineers together. They can develop their own simulated

platform. It means like, in terms of software interactions, the key point is how my module is going to integrate with the software module, which is the hardware specifically. But we are simulating that module integration at the software layer so that can be a virtualized platform.

So, I'll give you an example then how it is work in the industry. At HP, they basically converted their firmware in a single repository. Then they were able to simulate the printing engines as a simulated platform. And they were able to run millions of times. So, think about use case. Are they going to hardware for testing purposes a million times printing the papers versus running a virtualized platform and standing up as on demand and work around it? So, what we did, in this example, we carried out the hardware integration to the left as possible to shift left, then integrate it as quickly as possible. So, while the hardware engineers, they can work in the hardware component, but endpoint is ready for the software developers. And software developers will communicate that integration pieces.

Presenter: Excellent. One for Eileen, let's go to one from Frank asking, "Are there any best practices for when to use Kanban instead of Scrum?"

Presenter: Let's see if I can find my Kanban slide. So, at a high level--

Agile at the Team Level - Kanban

Similar to Scrum in the sense that you focus on features as opposed to groups of features

select, plan, develop, test and deploy one feature before the next feature.

Principles:

- Visualize workflow,
- Limit WIP, under the "in progress"
- Pull work form column to column,
- Measure Lead Time: Monitor, adapt, improve

**010 Scrum-- software engineering teams that use Scrum to organize their work are operating on a two to four-week time box, two weeks being pretty common. And so, the work has to be discrete enough that it fits nicely into that kind of time box. Kanban approaches can be really effective when things don't size that way. Think about things like research-oriented tasks where we might have to go out and study. We might have to develop a few algorithms and run them through some test data to determine which one we want to pursue. We might not be able to break that down into discrete tasks that fit into a two-week timeframe. But we do know we don't want to take on more work than we can handle. So, Kanban gives us another way to still limit the amount of work in progress. But we're just focusing on only taking on one, two,

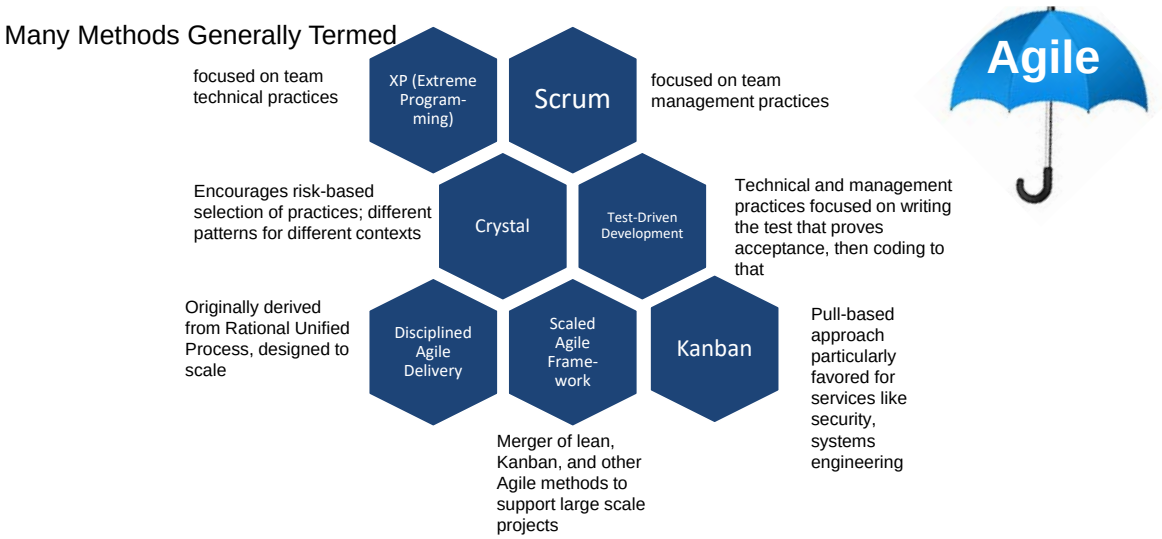
three things at a time. But we're not necessarily boxing them into that two-week conclusion. If we've got teams that are organizing their work this way, working with teams that are using more of a Scrum approach with that two-week time box, we can still keep a communication cadence between those teams at that two-week marker so that we understand what the status of the work is.

It's also anything that can be sort of service-oriented in terms of a business service or maybe like things that involve ticketing, like a lot of our military systems in sustainment will get a defect in from the field that enters a queue, and those defect reports are addressed almost on a ticketing basis. And Kanban approaches are really effective for that when the actual development activity doesn't happen in a two-week window. But it happens based on-- usually based on a release of funds or an arrival of something from the field. So, I hope that's helpful.

Presenter: Okay. How about one for-- let's go back to Hasan. "How does Kanban and DevOps link together?"

Hasan: So, going back to the relationship with Agile and then DevOps thinking, so the Kanban is one of the techniques of the Agile work.

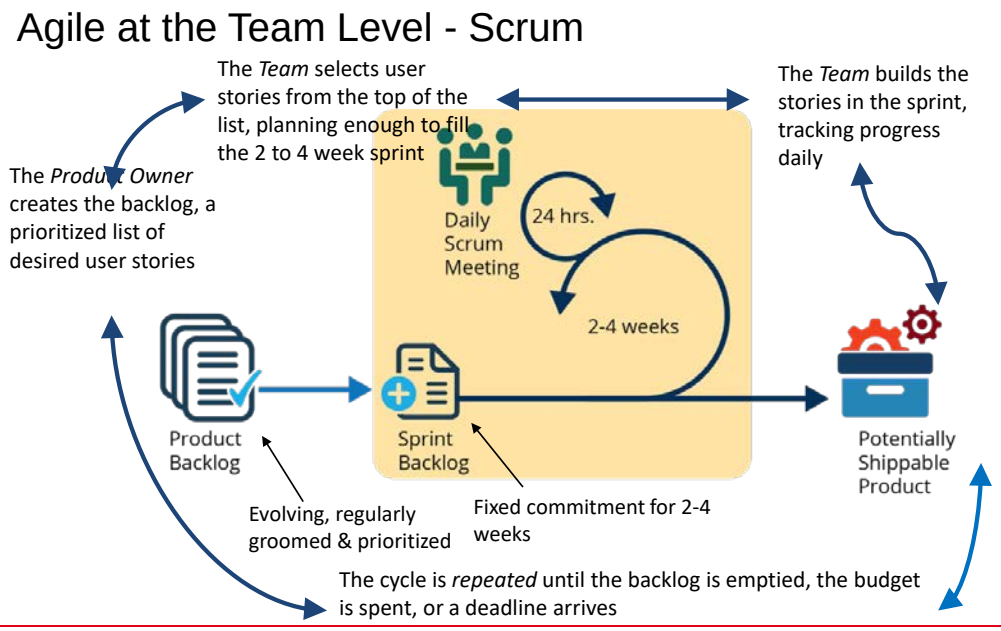
Agile



Common thought: Any method that is an alternative to documentation driven, heavyweight software development processes is probably an Agile method.

**008 So, it is a subset
of the Agile process is either Kanban
or Scrum. Then when you
put the work in place by either using
Kanban or Scrum, specific Kanban
you are talking about--

Agile at the Team Level - Scrum



**009 Giving the features--

Agile at the Team Level - Kanban

Agile at the Team Level - Kanban

Similar to Scrum in the sense that you focus on features as opposed to groups of features

select, plan, develop, test and deploy one feature before the next feature.

Principles:

- Visualize workflow,
- Limit WIP, under the "in progress"
- Pull work form column to column,
- Measure Lead Time: Monitor, adapt, improve

**010 It's going to get developed

from the code from the developers, and carry out that features and build it as part of the CI server, and deliver it to the staging environment based on that features. So, DevOps is going to give the capabilities taking that the Kanban features that has to be done, either let's say to do things, which is occur in the program-- people are going to do, and doing which is right now, the people are working on it, which is developers are working on it. And getting feedback from a user for a given feature set. So, as Eileen said, if there is no time box specifically, if you're focusing for a feature specifically to deliver it to the users, it's a relationship how you do the Kanban and DevOps together. But again, DevOps is not looking for specifically how you do get the features in place. Sometimes, you can do Scrum-ban, like I see some organization does a Scrum-ban, which is a little bit Scrum things, and having daily activities and putting it through a sprint but putting the features in it as the Kanban features into either the backlog or to do, and then carry out to the DevOps process. So, as long as if you are able to trace the work, which is either through the Scrum or Kanban, and put into your CI environment, and map that changes back into your case management, which is typically Scrum or Kanban things you're doing, and have a feedback for that work has to be done.

So, the feedback is the key point here. Feedback is getting from users directly. Or feedback is getting from

operational people directly. So, get the feedback for the given features. So, that depends on how the teams are agreed on it and how team is going to work, maybe the Scrum or Kanban. But the team has to decide what type of process are we going to use. Put in the DevOps pipeline, get feedback throughout the pipeline.

Presenter: And I think it's important to know that there isn't a need to be a perfect Scrum model, or a perfect Kanban model. It's really about limiting the amount of work that you're doing at any given time so that you can finish everything that you've started and get that rapid feedback from the users. So, really whatever works for a development team, whatever is understood by that whole organization, it doesn't really matter whether it's Scrum or Kanban at that point.

Hasan: But the one thing that I would like to suggest that it doesn't mean every developer has to do its own thinking. It has to be a collaboration and communication amongst the team members. If the team are working an application or a product, that team has to use the same process. If they're using a Kanban, everybody has to use the Kanban.

Presenter: Right.

Hasan: I cannot use the Kanban while Eileen she's going to use the Scrum because it makes sense.

Presenter: Right.

Hasan: We have to be on the same page to make it a better product.

Presenter: Okay. We had a question come in from Junior asking, "With Agile and DevOps, development, release, and deployment are moving faster and in a more transparent way. Does this initiative become prone to insider threat or is it going to bring more value to mitigate it?"

Hasan: It is going to bring more value. It looks like it is a threat. It looks like you are getting so transparency across all of the stakeholders, all the projects, and including dev and the sec folks. But all team members are on the same page in terms of the business goals and responsibilities. So, look at the insider threat. Threat is maybe somebody's malicious work, and they are using it. But if you are putting a true DevOps pipeline, you can measure any type of bad activities throughout the pipeline because it's going to bring traceabilities.

So, think about it this way. If I'm doing the something bad by myself, it is not exposed to another person, I can much easier. If I do something bad, if it is transparent to everybody, it is going to be very difficult, maybe impossible, because if I do something bad, people are going to see that. And also, if I'm doing the code checking for the code I'm working on, then if I'm doing some testing activities, all the test results are

going to be seen as publicly-- dashboard publicly means within the team members. The team can see the progress when I'm going it. So, we can eliminate any type of insider threat activities making it more visible, more transparent. So, bad things cannot be happening. If then something happens, we will see right away.

And putting the right checks and balances in terms of the development pipeline, like who's going to do the code reviews, or who is going to do the code checking things, put security controllers in it maybe using SSH keys? These are techniques we can talk about more, but once you put the full integrated pipeline with the controllers, it's really easy to find out what is really happening behind the scenes. And like administratively, if somebody is going to go change some settings from-- as part of infrastructure as code service, maybe pushing new updates, maybe putting some other backdoor things, it is going to be transparent to the rest of the other admin folks as part of infrastructure as code pieces. So, basically, we are writing a code as a scripted whole environments. So, we are eliminating any type of insider threat, technically, using DevOps principles.

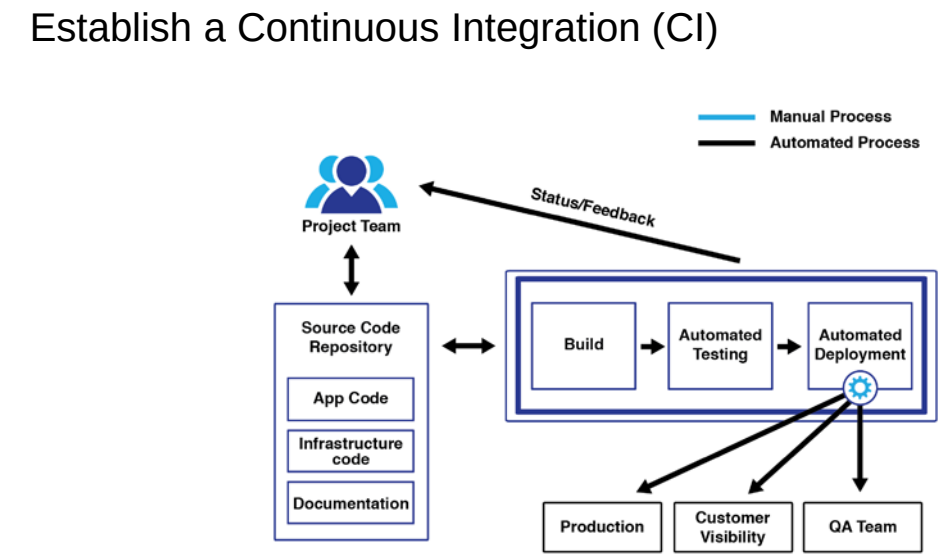
Presenter: Eileen, anything to add there? Or we can move on to the next?

Presenter: I think we should move on.

Presenter: Okay, so we've got one from William asking, "Which data display techniques/charts/tools seem to be the most preferred/effective for the teams for communicating status progress to senior leadership?"

Hasan: I'll take this one as well. So, in the DevOps pipeline-- I'm going to go to the slide, actually. It's going to be much better to explain. So, this is basically a model for--

Establish a Continuous Integration (CI)



**020 Continuous integration. So, if you look at that slide, and we would like to collect the data from various phases in lifecycle. So, what are the various phases are in collecting data during the build process, that's one effective way, and how many cases we are working on it, and how many cases has been built successfully and show the result, either maybe a pie

chart or maybe just a build result, maybe just to put the kind of like a grey balloon, kind of like a lighting things we can do. So, an organization can pick out as many visualization portals available as an open source or the product-wise. But key point is here to get the data as part of the process, as end of the build process. Or if you are deploying into the staging environments, you are able to deploy successfully.

Or if you are adding some certain test as part of the build process, especially for secure testing, are we passed the test? Where are we in terms of test, which is going back to the feedback to the team? Then we passed it as the build-wise and test is successfully done. Build goes back to the feedback to developers. It's done, or maybe security folks as the next step.

So, other things-- and see the progress in development activities. How many cases we have, which is going to be part of your issue tracking system, then see how many cases have been opened for a given sprint frame. Then you exactly what is your workload. Then other things you can do, see the overall load for a development team. Let's say I'm working for ten tests, but another developer only working for one test. It's not fair to have that work distribution. That's another way you can get metrics for more about development perspective.

So other metrics is more performance related. Say if I'm putting an

application into my production environment, then I can measure many things, measure things like-- it can be a CPU. It can be a network traffic. It can be data users. It can be latency. There's many things that can be implemented as the application health, which another feedback to the team, which is more about the how I'm doing in terms of quality of the work. Is it good quality? Or where I am in the quality? That metrics can also be defined from the QA person or defined as a team together. What are we going to measure it? And what are the data we're collecting?

So, another example like the Etsy is a good example from industry. They were able to monitor full pipeline. And in some cases, they were able to pull two thousand-- two hundred thousand type of records from their systems for various components of the systems. So, how you visualize it, again, depends on data, maybe a pie chart, maybe a graph. And it depends on your data visualizations.

Presenter: Okay, let's go to one for Eileen. Susan asks, "How does this apply, meaning Agile and DevOps, to weapons systems? It makes sense for IT, but I don't see how it applies when you're, say, acquiring airplanes."

Presenter: We get a lot of questions like that. How do I spin up the wing of an airplane on a two-week cycle? So, software is capability. Much of the capability that military systems,

other embedded systems, deliver is enabled by software. And so, if we think about all of the various software elements of those systems, we can iteratively and incrementally and in small batches successively approximate towards the capability that we're trying to deliver. And these approaches give us a lot of flexibility to exploit changes in the technical environment and respond to changes in the demand signal of the capabilities.

So, we can do things like, in a weapons system environment, we can even do things like we-- model-based systems engineering allows us to really apply small batch iterative validation to various-- even various designs for the hardware. We're doing some work now on even continuously virtually integrating aspects of the architecture. So, we can do a lot to continuously integrate, build, and test software to discover software to software integration plat--

DevOps has four Fundamental Principles

DevOps has four Fundamental Principles

Collaboration: between project team roles

Infrastructure as Code: all assets are versioned, scripted, and shared where possible

Automation: deployment, testing, provisioning, any manual or human-error-prone process

Monitoring: any metric in the development or operational spaces that can inform priorities, direction, and policy

**018 Integration problems before we actually marry up the software with the hardware platform. There's a whole lot of space here where Agile and DevOps principles apply even though we're not talking about a two- inch slice of the airplane every two weeks.

Hasan: That's a good point, actually. Sorry to interrupt you, Eileen.

Presenter: No, go ahead.

Hasan: People are misunderstanding, especially in the other space. It doesn't mean that we have to deploy every minute or every day. But there are some principles we can use, Agile and DevOps principles, to make it much better work in terms of eliminate any rework. So, if you get early feedback for any module as being part of the airplane or warfare systems, if you get a feedback for

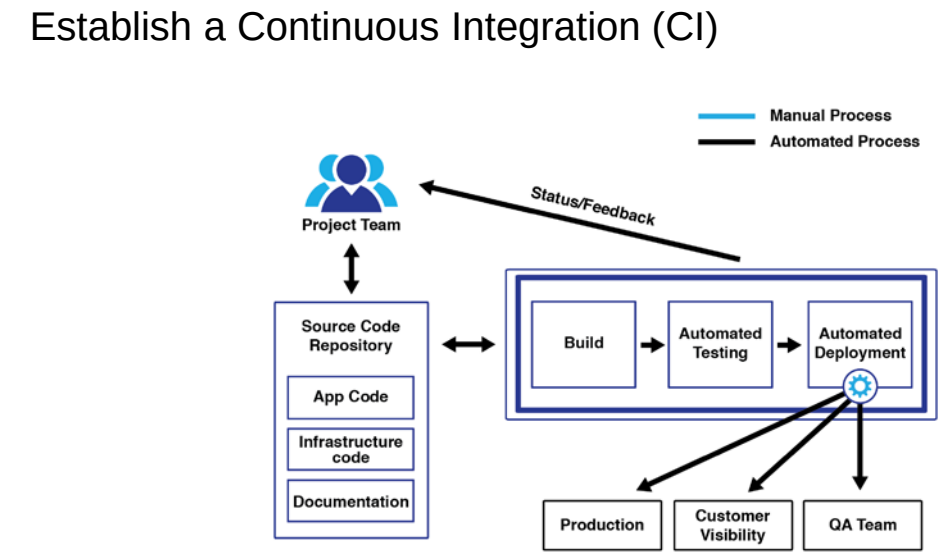
that modules early, you're basically saving a time of that development activities. Ideally, we'll get a six month or you can be a year later. Then you can get feedback.

So, other things also, the one of good princ-- I like it for continuous delivery. There's a continuous delivery, continuous deployment. Those are two distinct things. So, continuous delivery is more about delivering a code into your staging like a production-like environment. And think about the Navy cases, a shipyard. So, a ship comes to the yard every two years. But it is only one small window that we have to do all the deployment in the small window. So, you going to make sure it's going to work time. But you're going to have to practice with DevOps practice. Then you can do continuous delivery multiple times. Then you have assurances when the ship is available for deployment for a short timeframe, it is going to go smooth as early as possible because you have a very short time to deploy. So, taking that, the concept of continuous delivery, make it practice a lot, hundred times maybe, depends on how much you do. If you do it a hundred times, it's guaranteed that hundred and first will be successful as a continuous deployment. That's the benefits of DevOps, actually.

So, take that principles and how we employ into the weapon or airplane or the hardware system a bigger picture. Then use that principles. So, one or two principles I like for

traceability perspective and how you get that tracing requirements all the way to the production-like environment or production, which is DevOps is bringing other traceabilities because we said everything has to be scripted, or everything has to be automated, or everything has to be part of the--

Establish a Continuous Integration (CI)



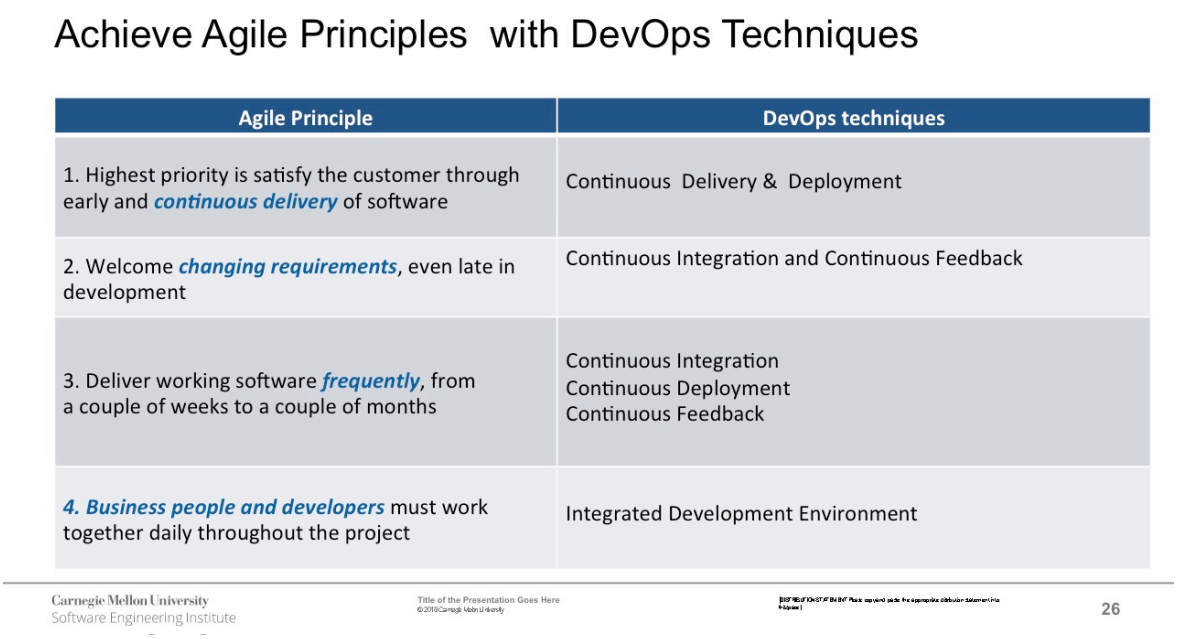
**020 As the repository system, including documentation, including test cases. So, basically, eliminate any type of communication problem amongst the team members. And there's a single place people can go and take a look over there, including project managers and QA and testers. Everybody has the same portals all sharing together.

Presenter: Excellent. All right, back to a question from Mark. He said, "As

you probably know, the DoD has independent testing requirements. How does this help us with that? We often have months-long bottleneck when we hit operational testing."

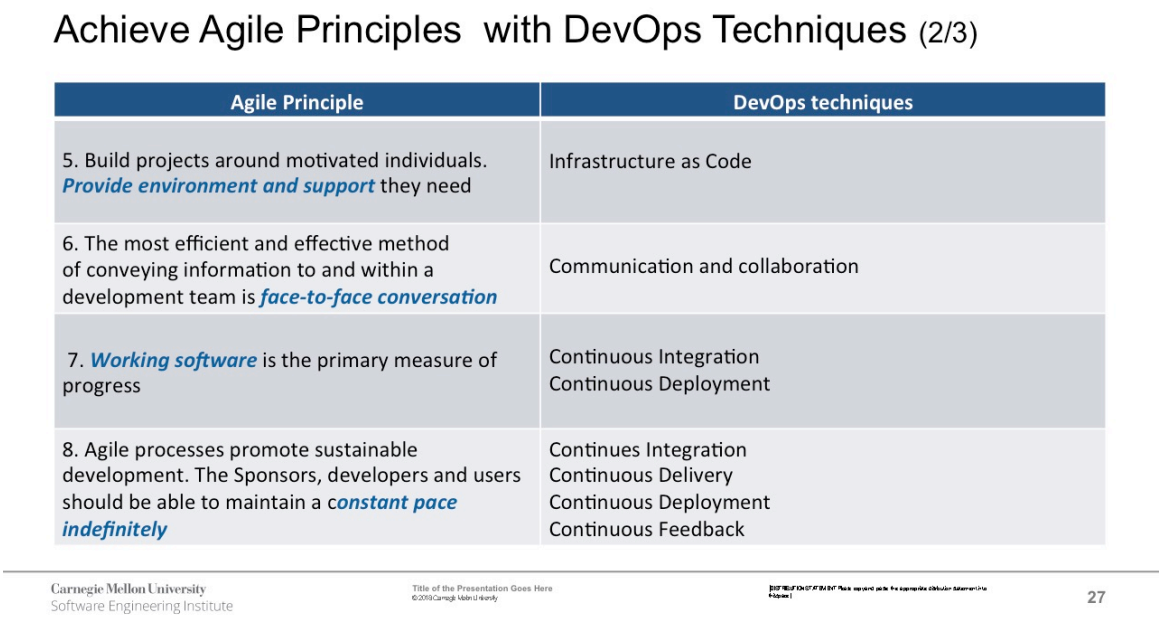
Presenter: So, the reason that we get those bottlenecks is often because we've had delays in programs. And then we basically throw everything at the test organization all at once and say, "Here, now you test it. And now, we'll go away." OSD is actually working to push the engagement of the operational test community a lot more to the left--

Achieve Agile Principles with DevOps Techniques



**026 Getting involved even at the requirements level. So, that collaborative approach-- that collaborative approach--

Achieve Agile Principles with DevOps Techniques (2/3)



Achieve Agile Principles with DevOps Techniques (2/3)

Agile Principle	DevOps techniques
5. Build projects around motivated individuals. ***Provide environment and support*** they need	Infrastructure as Code
6. The most efficient and effective method of conveying information to and within a development team is ***face-to-face conversation***	Communication and collaboration
7. ***Working software*** is the primary measure of progress	Continuous Integration Continuous Deployment
8. Agile processes promote sustainable development. The Sponsors, developers and users should be able to maintain a c***onstant pace indefinitely***	Continues Integration Continuous Delivery Continuous Deployment Continuous Feedback

**027 In having those stakeholders involved from the start to help understand-- A, understand what the requirement sets are and eliminate any confusion, that reduces a lot of variability when we get down to those-- what we call those big bang test activities. We also, if we're using Agile approaches and DevOps approaches, we're successively building a body of evidence that what we have built works. So, every two weeks, four weeks, six weeks, whatever, we can say we can have a report available for the test organization that says this is what I did. And it works. This is what's going to be coming at you at that next big bang test event. So, basically, we reduce variability in the quality of what arrives at the test event. So, there's no mystery. And these should be-- these should wind up being snooze-fests rather than

opportunities to discover a lot of problems because we've been doing this continuous integration and continuous delivery all along. And I'll let Hasan add any other thoughts.

Presenter: One more thing added, if they're working as a silo, that means if the tester groups are working by themselves, not working talking with the dev or not talking with the users directly, it's going to fail because Agile and DevOps is you have to have a collaboration. So, you have to have early engagements. So, whoever is writing the test cases, and they have to share upfront early, and maybe work together with the users directly, or sharing the script what they wrote and share the script with the dev team or the CI team, so they will implement the continuous integration. We do not have enough time it seems, but we can talk more.

Presenter: Yeah, we've got about a minute left. So, I'm going to go just rapid fire, how about the shortest answer possible for as many questions as we can. We had a question from Tushar asking, "From an implementation perspective, can we make a model where Agile focuses on process and mindset, and DevOps is about engineering practices encompassing development, operations, and support, testing, and security?"

Presenter: It looks like a fair grouping things. But still, DevOps is requiring a little bit of cultural changes as well. So, if you are able

to get the cultural changes as part of Agile process, I'm okay with that definition.

Presenter: Yes, me too.

Presenter: From William, "Which data and metrics seems most preferred by QA engineers working with DevOps teams?"

Presenter: So, the specific-- before again, going back to the-- I think we talk about a little bit about the visualization pieces. So, metrics, you're looking for the-- how much the fact we are fixing our-- throughout the DevOps pipeline and how much data we eliminate for rework. And the key point is you don't want to do the rework, basically, measure that, and also measuring for the mean time to recovery, MTTR, failures and other techniques to measure the datasets, and looking for the-- how much ticket we have at the beginning and opened. Ticket means like either support, maybe open cases, and how much we solved using DevOps techniques, and eliminating any reworks. So, that's another technique where the data has to be de-riven.

And in terms of security days and how much vulnerabilities or deficiencies in the program as they exist, eliminate the deficiencies based on the automated secure testing. You can assure that. And I had some such and such deficiencies at the beginning. And I eliminate that as a result of the feedback as a report.

So, the other metrics can be collected. But specific metric has to be decided as a team together. What are our business goals? Is it effective? What are my security goals I had to achieve? So, that metrics should be decided through the business and through the team together. But these are things I can suggest definitely you should have it. And you can add more on top of it.

Presenter: Anything to add there, Eileen? Or are we--?

Presenter: No, that was great.

Presenter: Okay, so it's about two o'clock now. I just wanted to give a quick plug for a symposium we have at the end of this month in Washington, D.C. It's the Software and Cyber-solutions Symposium. And the theme is DevOps and Agile. So, if you like what you heard today, this is going to be right down your alley. Again, it'll take place in Arlington, VA. Eileen and Hasan both are talking. So, you want to give a quick plug on what you guys will be speaking about at the symposium?

Presenter: We're really both going to be talking about how these approaches aren't just limited to just the bits and bytes. I'm going to be talking about how they can be leveraged across the organizations into services and how management behavior fits into that. We're going to be doing some tutorials on metrics for program oversight for DoD executives and Agile readiness and fit

looking at organizational pitfalls and risk in investment for pursuing Agile adoption for DoD and government executives. Hasan?

Presenter: So, I'm delivering a tutorial on the first day and on the DevOps, more about the management things because it's also requiring some cultural changes, especially in the regulated, highly regulated environment like DoD, and what type of things the managers they have to know. Today their job is better in DevOps way. Which is I'm going to give tutorial four hours.

Then the talking, I will deliver a talk about implementation barriers. So, it looks like it is very hard to implement. But I will show it is not that hard is implement. So, I will have a talk on that what are the barriers and enable us implementing DevOps properly and including security concept. Also, we're going to have a panel and sharing the best practices from industry and also DoD and how DoD did it, how the folks in industry did it, have come in together and share their experience including security and how the DevOps is enabling to include security into the process.

Presenter: And folks, I left out the best part. It's free. The symposium itself is free. There is tutorials on the 26th and 28th that are also free for .mil and .gov attendees, a small cost to industry, but we'll send out more information on the symposium. Probably this evening, you'll get an

email with the archive to this event and information about the symposium. So, we hope you can attend.

Tons of questions we didn't get to, folks. We apologize. We probably should have done this for an hour. But thirty minutes, we got through as many questions as we can. So, if you have questions that weren't answered, feel free to reply to the follow up email we send out this evening. And we'll get you an answer from Hasan and Eileen.

Presenter: Come to the conference.

Presenter: All their free time.

Presenter: Or come to the conference.

Presenter: Come to the conference.

Presenter: Come to the conference, even better. And you can ask them in person. That's all the time we have for today. Our next event will be next Friday, March 9th.

Contact Information

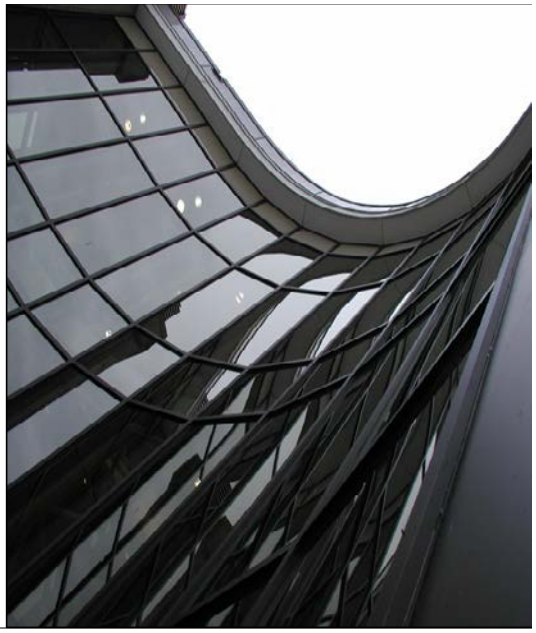
Contact Information

Eileen Wrubel
Technical Lead,
Agile in Government Team
eow@sei.cmu.edu

Hasan Yasar
Technical Manager,
Secure Lifecycle Solutions
hyasar@sei.cmu.edu
@securelifecycle

Web Resources

http://www.sei.cmu.edu/



**030 We're going to do, at 11:30 to 12:30, an event called Three Software Innovations that the DoD Needs Now. And we hope everyone can attend that. Have a great day. Eileen, Hasan, thank you very much.

Presenter: Thank you. Thanks for having us.

Presenter: Thanks, everyone. Goodbye.