Shane McGraw: Hello! And welcome to today's SEI Webcast, DevOps Enables Digital Engineering. My name is Shane McGraw, Outreach Team Lead here at the Software Engineering Institute, and I'd like to thank you for attending. We want to make our discussion today as interactive as possible, so we will address questions throughout todays talk, and you can submit those questions in the YouTube chat area, and we will get to as many as we can. Our featured speakers today are David Shepard and Hasan Yasar. David is a Software Developer in the Software Engineering Institute's Software Solutions Division, and David has made a career working in many different areas in the information technology field. Hasan Yasar is the Technical Director of the Continuous Deployment Capability effort here at the SEI. Hasan leads an engineering team group that's tasked with developing prototype solutions within DevSecOps. Now I'd like to turn it over to Hasan Yasar. Hasan! Good afternoon, all yours!

Hasan Yasar: Thank you so much, Shane. Good afternoon, good evening, wherever you're joining. We have worldwide viewers all around the world. Thank you for attending webinars. It's going to be an interesting day today, because we're going to try different technologies. We will use the whiteboard in the more interactive session. I generally encourage all of you and please ask the questions in the chat window. I wish we had it in person we can attend and then talk, but please use the mechanisms like we all used to have a virtual settings, called YouTube, and put your request into the chat room, and we'll be monitoring. I will let the team respond to questions, be more interactive. So, what I'd really like to talk about today, and we'll be discussing DevOps principles or DevOps guidance and how it's going to help Digital Engineering. So, what we would like to do today we said, "Okay, let's take a look at the Digital Engineering, let's take a look at DevOps piece, and let's see what its commonalities are. What the challenges are, what the perspectives are? So, we came up with an agenda, and what we really discussed in this agenda item, we're going to go over what the DevOps means first. And then we're going to talk about Digital Engineering, what the Digital Engineering means, and we will go over to the goals. And then we'll cover DevOps goals and Digital Engineering goals. And we'll look at challenges, what the challenges are. We may see the challenges, and then we're going to see how the DevOps is enabling all those challenges. And always the question is as toward the end of the webinar, we're going to discuss what is next? So, how can we combine these two elements together with the kind of next things we should do for the implementation perspective, maybe, make it more reality perspective, and then where to start? But as I said, please ask your questions in the chat window. That's the first thing we should really focus on to break the silos, break the culture. Culture really start from asking the questions first. So, before that, I'm going to dive into the what DevOps means first, and then I'm going to hand it over to Digital Engineering. So, what does really DevOps means? I know you already heard many times with respect to the DevOps, what DevOps means, but let me go over one more time what the DevOps means. So, at this we had to set the stage. So, DevOps is the kind of set of practice and also the process that enables the communication within all stakeholders. See it's not that DevOps, it's basically other people, other people or other stakeholders. So, we would like that all the stakeholders together and are working with respect to the integration. So, there are four principles, and one is the collaborations, which is in creating environments or creating cultural piece is of that the people are communicating across the functional. It's not a single team a cross organization team, which is the first element is the collaboration perspective. And then the second one is infrastructure as code. This is really fundamental concept of DevOps. It is very relevant to the Digital Engineering. We're going to cover that shortly and talk about what is the infrastructure means in this context. So, everything, all the assets should be versioned, scripted and shared where possible. Like shared is the key point is we have to share all the artifacts with the stakeholders. So, everything that we are building that can be your test harness, that can be your platform, that can be any work, it has to be versioned, it has to be scripted. So, we can increase the automation, because if you have the scripting that you have done, if you build up the build scripts, if you build up your model script and example, you can automate those and then you can automate testing, you can automate deployment, you can automate various component in the pipeline perspective. Then last one is the monitoring. It's really monitor how we are doing well. This is another relationship again we're going to cover up in detail how that really monitoring metrics is really helping this engineering perspective. Like monitoring is collecting metrics. I said any metrics, actually metrics should depend on your organizational context, what needs to be done with specific metrics. And put into the development or the lifecycle and why so we can monitor those metrics and also at the same time, we can visual those datasets to check our self. Are we doing well? What are we doing in terms of the deployment? What is our direction? So, we can more uniform and more informed decision based on what we are like to do. There are a few challenges, let's take a look at the challenges before diving into the Digital Engineering. So, overall, DevOps overview perspective, I know we set up common principles, but we have a cultural pieces playing a very key role in implementations. We have a technology aspect, which is a tool, what type of tools are we going to use? So, what tool's going to be helping us to build up the automation perspective, infrastructure is kind of related together, technology and tools. We said about the communication so we had to build up a good communication mechanisms between the stakeholders, such as using the common platform, maybe using some other chat in a while, or maybe using some Wiki pages. Whatever the communication mechanism are, it's going to help us to build up the good shareable, infrastructural, but also we can learn. There's a learning pieces as well. And two things about the people. People are us. So, we are the engineers. And we are the people. We are the developers and architects and then system architects or operational team, security team members are really, really working to get the things done, which is building up, designing, modeling, verify, validated. Make sure it is a security. So, all these people actually, we are them.

So, it's a really crucial element. Tool cannot do anything by itself without having the right people. Without having a right skillset where the people can achieve the work we have to do. Which we have the processes really supporting what needs to be done. What are other-- the lean practices and processes are we can use so we can combine all these elements together so we can build up a DevOps in this section altogether. So, I'm going to pause here, and hand it over to Dave, so let's look at Digital Engineering now just to have some foundational background for Digital Engineering. David, what do you think about Digital Engineering, so I know we talk about DevOps a little bit. What is your thought of Digital Engineering?

David Shepard: Thanks, Hasan. So, Digital Engineering is a process model that is-- shares a number of similarities to DevOps, DevSecOps, in that it brings together people, process, technology and the data about that product that you're building. It's a process model. It's defined as an integrated digital approach that uses authoritative sources of systems data and models as a continuum across disciplines to support lifecycle activities from concept through disposal. There's a lot in that definition, and the part that I would probably talk about first is that authoritative source of systems data. So, in-- it's not necessarily common for all of the engineers and all the other stakeholders that are working on a project to share all of their data at all times, it's all available and they're all working from the same source. That shared source of truth brings with it a number of benefits to the speed and the ability to iterate designs throughout the lifecycle. Yeah.

Hasan Yasar: David, you're talking about there we have lifecycle, that's kind of striking me as well, the DevOps perspective. So, we have a lifecycle and the lifecycle is not just a software perspective, right? Lifecycle is about the Digital Engineering perspective. Maybe we can look at the-- at goals. Probably we can go from that model, actually we have to-- that model was created, a Mrs. Zimmerman, and as the DOD Digital Engineering Strategy, which is from 2018, and she described the various goals. Maybe we can look at each goals and see which one is going to be enable, what needs to be done to achieve these. Because there are some challenges associated with each of goals, right? Well, I'll begin with the Model Integration, etcetera, some other things we can look at that. So, as you said, there is a lifecycle. So, now we are talking about the lifecycle, not just the software lifecycle first of all, right? You are talking about the hardware or a system lifecycle. Can you open a little bit your-- the lifecycle thinking, David? Like when we say-- I know typically we know the software lifecycle, now we are talking about Digital Engineering perspective of lifecycle, because that's specifically talking about end-to-end solutions right here, right?

David Shepard: Definitely, that's right.

Hasan Yasar: What is your thoughts on lifecycle

David Shepard: That's right, so we're talking about a product lifecycle, something that is a integration of many different-- potentially many different hardware components as well as software. And all of the-- how you bring all those pieces together and manage the lifecycle of that product over time as it changes. In the case of a lot of the systems that our customers buy and work with, those systems are kept around for a long time. So, they have to evolve those systems over time. And it's important to be able to know what the changes have been to those systems over time, and why they were made. So, the idea of product lifecycle management, it isn't a new concept, especially in industry, but it is, you know, Digital Engineering is more than that. It's an expansion of the idea of product lifecycle management to incorporate modern digital techniques across all domains and stakeholders of the project.

Hasan Yasar: That's actually a good way-- there is a question from Jim, and he's asking about, "Can you address how this applies to your product line? A panel of technical similar system as well?" What you describe is a product line, right? So, we are not building just the software, we are building the hardware. And hardware and software goes together.

David Shepard: That's right.

Hasan Yasar: That's the system we are building.

David Shepard: Yeah.

Hasan Yasar: So, let's look at the-- oh, go ahead, David.

David Shepard: No, go ahead.

Hasan Yasar: I said maybe we can dive into the-- what is the goal of Digital Engineering? So, now we can list some challenges of each of the goals we would like to achieve. So, we can dive into that and see how the DevOps is really helping or enabling to overcome those challenges. Maybe you can start from the model integration, or the first call from Digital Engineering strategy concept.

David Shepard: Sure. So, I think it's important to recognize that the strategy that we're going to look at today was designed by our particular customer base, and everybody who's implementing something like this is going to have a shared-- you know, a common set of challenges that they have to deal with. But everybody's strategy is going to be unique to their organization and to their product goals. One of the key things that our customers deal with is they don't get to come at a lot of things greenfield. There's a lot of legacy, a lot of existing process and technology involved. And a lot of those products have been designed with a focus on thoroughness and minimizing risk in developing and delivering those products. And one of the challenges that that focus on thoroughness and minimizing risk creates is a challenge with your ability to deliver a new capability fast, and iterate on things, and respond and adapt to a changing world. So, you know, for our customer, this is really about being able to adjust faster, and change in ways that are relevant in a timely manner. So, for the first goal, you know, formalizing models, the development of them, and how to manage them, this is-- the challenges for our customers, our stakeholders, you know, they're developing models to support decision making and engineering activities, and they need those models to reflect the needs of that decision-making capacity.

Hasan Yasar: So, David, when we look at the models, I think the results of the questions relate to the hardware pieces. That's kind of like a core element of the Digital Engineering concept. We have to start from the models. So, model will be one of the component of the system, right? So, when they build up the model, there is going to be integration challenging, which is having an integrated model into the product lifecycle. So, we don't want to be integrated model in the late in the lifecycle. We would like integrated model early, early as possible in the lifecycle. So, let's say if you are in the planning phase, probably, this is kind of our operational phase, we have to really start to integrate the model as early as possible in lifecycle. Early means like the-- maybe planning session. Maybe we have to do that before the planning session. So, this is the similar concept as reminding me in the DevOps perspective, so we have to really get the early feedback from our stakeholders. Early feedback things we are building, or things we would like to be-- get some idea from stakeholder, it can be users, it can be whoever the stakeholders are. So, in a typical DevOps world we define the-- we say the sprint. We say we have a sprint, so if they're following the agile methodologies, maybe a scrum, maybe scrum], whatever methodologies people are following, within the scrum, let's say we have true experience we do. With a true experience, we will not be shown something is kind of what we have done the true experience. So, going back to the model, maybe we can propose that model integration can be part of our experience within our pipeline of the infrastructure, try to validate early design of-- we're not really expecting the full complete model, right, Dave? But expecting something is early prototype, maybe early proof of concept, whatever the first creation of the model, we can really start to get verification, prospect-- or the validation yet, at this DevOp for verification perspective right here, and then how we are dealing with the model and how we are designing the system. Try to get early feedback from stakeholders. Same mentality with the DevOps, right?

David Shepard: Exactly, exactly. So, that's where we-- one of the places we start to see similarities between the process models is in the area of the ability to iterate quickly and to improve on designs. If failures happen, they're going to happen. You accept those, you learn from those failures and you move on with your development process. The focus on, for example, SpaceX is a good example here. So, they build rockets. And you know, rockets are notoriously fickle. They blow up all the time. If you focus solely on making sure your rockets don't blow up, and you do that, you know, it's kind of a premature optimization if you're trying to get something delivered quickly to spend all of your time focusing on the reliability of the rocket, before you have your design completed. So, they take this approach. They have a set of goals for where they're trying to get to with their designs, and they iterate quickly and make adjustments and test those changes and collect data and learn from their iterations as they go. So, that's how you get to this place of quicker capability delivery is by iterating on your designs and not trying to engineer the entire solution upfront, because you know, we know from decades of experience that that doesn't work real well.

Hasan Yasar: So, as you describe, I think the SpaceX was a good example, it was really talking about changing the perception from meantime infant failure to the meantime to recover basically. So, that's kind of how-- it goes back to the learning. There is a question on the chatroom as well, "How can we really continue to increase our knowledge base?" We have to create a collective data, what we learn. And then I remember that as well, it would say specifically the most, we learn what we would like to get at. We got the data. So, we learned what the model iteration it is. We got the model. We got the data here. That data is going to help us build up our knowledge bases eventually. This is kind of like a how really mindset changing and try to put into the lifecycle. There should be mechanisms to collect the data what we need, and then try to recover from the data so we can get better the next time. And that goes back to our product lifecycle perspective. Let's deliver something. Let's look at the model. Let's see how it is behaved, that model, we need it running already in the test cases possibly. Or maybe just building the component. So, every iteration is creating some datasets for us. Right, David? Every iteration.

David Shepard: That's right.

Hasan Yasar: It can be just running a test. It can be just building the model, verifying with the users. Maybe doing some other analysis we do in the same environment, whatever we do, it will generate some datasets. And I'm going to take the note, each of these goal technically is generating a data for us, each of them, right? So, this data will help us to build up a knowledge base eventually. So, what's your-- anything you added, David, here?

David Shepard: No, the focus on the data is really important. You know, you're collecting all of this life cycle data and you're using it to inform decisions and design changes and iterate on your designs. And you know, it helps you do things faster. But when you're talking about large systems, they have to do integrations and all of that data, by collecting it and sharing it and having everybody have the same set of models and data, they can make those iterations faster, and they can still achieve their integrations. So, modeling helps with that and using the data with those models.

Hasan Yasar: I guess it's going to give others also reusability the perspective, right? It's going to give the usabilities for the scalability perspective-- we're going to cover up shortly about the scaling the systems. So, I'm going to jump to the Goal 2, and for Goal 2 about the Authoritative Data source, which is how we're going to get a typical Authoritative Data source, so your data source. What do you think about the challenges here, Dave, with your experience?

David Shepard: All right. So, this is another touchpoint where there are similarities between DevOps and DevSecOps and Digital Engineering. The need to share data and have everybody working with the same data source code, models, whatever it is, is a very desirable thing. And while there are a lot of similarities, there are actually a lot of differences in the way the tooling at the technology level supports integration and the way people are using the models and sharing them. So, for instance, a typical version control system in software today is distributed version control, something like Github]. And that distributed version control system basically gives everybody their own copy of that repository of system data that they work from locally, and then make their changes and then push it back to a shared repository when they're ready to sync up. That process is-- looks a little different when you're dealing with large assets that don't-- and the need to run simulations that you can't really expect people to be able to work with them on their local hardware. So, the tooling is a little different in the way you integrate the tools that work with your models is a little different, but the overall goals are similar.

Hasan Yasar: So, the foundation concept here to overcome that challenge is really building up a infrastructure, that infrastructure that we are sharing all the data. So, keeping all the artifacts into the common repository management, an example. Keeping all the security control, there is a question on the chatroom as well, the security question consideration, yes! We should get a consideration of security, how we are creating a model right here, but where we are putting all the security requirements, we have to put into the same repositories. All of these artifacts should be the similar repositories. So, another concept in the DevOps perspective, when we say infrastructure as a code, or where we talk about artifacts, we're not just saying the code itself. We are looking for other elements together. Other elements are the design documents probably you may have. Other elements are the build scripts. Other elements are what we did for the specific components, put all the artifacts into the same code repositories, keep the same versioning of the repository, which is kind of a sharing. Big challenge for Digital Engineering perspective, there are very multiple pipelines we may be setting up. There are various subsystems we may be setting up. How are we going to share information, official information in a work documents, or in kind of a static documents, through the email? Which is just going to keep losing the track of the what version it is, where we are. So, it's really challenging with the same data that everybody can use that. So, DevOps will bring about a collaboration perspective, communication pers-- we say, "Use the same platform, or to use the same platform for the team members." Again, we're not forcing everybody in a single platform, but build up a product lifecycle. In the product lifecycle, we may have multiple product lifecycle, we may have it, right? So, each of the system component, they may have a System A, System B or System C that we are developing. So, each of the system components can have their own pipeline. We are sharing data here, we are keeping the same repositories, putting different data components, A and B platforms, C. So, technically, we are able to share each of the datasets in-between, A to B, or to C, or other component we can use. So, technically, we can have A's and B's and C's, we can use all of the component, we are sharing together. So, actually we are using the DevOps concept, or using the common repositories and using a common place and a common governance environment and asking or sharing the artifacts together within all stakeholders. This is the really key point for alternative data source perspective. Again, we are generating a lot of these artifacts. These artifacts are-- goes back to David's point, every steps is creating data for us. Every steps the product lifecycle. Let's say we have-- let's go back to the different context, let's say we are creating a model, so that the model we are producing a code, and that code, probably we are building the code, which building a model, and then we are testing that model. So, the person what tests goes, either goes to production or it goes to model. So, each of the lifecycle's generating data for us. So, each of the data has to be shared and stored into the common repository management that we are talking. This is how we can relate the alternative source of the data, that can be shared within the pipeline. Again, there's no single pipeline. We have multiple pipelines. Now we are talking of pipeline of pipeline concept, trading back to the various pipeline component, right?

David Shepard: So, Hasan?

Hasan Yasar: Yes, David.

David Shepard: Thinking for a moment about the security questions you had, "Are your security questions in consideration started early enough in the process?" To me, that question is one of culture, right? And culture and cultural change is a important aspect of this, right? And the successful adoption of any process model is going to require some amount, possibly a lot of a organizational cultural adaptation. And in the case of security, in the development of a product, you're talking about bringing in the stakeholders who have an interest and concern for security into the development and lifecycle management of that product. So yeah, it should be, if you're doing this the whole process, not just focusing on the technology, the cultural aspects should bring in stakeholders, like security.

Hasan Yasar: Yeah, actually, you remind me, David, and there was a survey done from the Systems Engineering Research Center and based on their survey data as number one obstacles came from the organizational culture as the 44 response was saying the organizational culture is the big obstacles implemented Digital Engineering model-based system engineering concept. The same true also DevOps perspective, too, which is more about the culture. When we open up the culture, David, I think it's more about sharing the information, change the mindset. That goes back to your example for the SpaceX example, right? Changing back to the-- learning from our failures, more about learning how can we fix the failures? What we are getting from the perspect-- like changing from, you know, concept of MTTBF and when goes back to the MTTR, and bringing back to the--

David Shepard: That's right, you have to get away from a culture of blame.

Hasan Yasar: Right, culture of blame and learning.

David Shepard: You don't want to decide the blame.

Hasan Yasar: Yeah. This is the same mentality for DevOps. What we are saying is they, "Let's learn from our mistakes. Let's learn from our data." But the sharing the information, actually creating a good feedback mechanism that maybe can open up a little bit here, and we need to create some sort of a feedback mechanisms, right? How we are talking with each of the elements in this example, like each of the components that we are building, each of the capability we are building, each of them is creating a data for us. You know, we have to share this one with the right feedback, right? So, we can share the feedback between each team member so we can learn from our mistakes. We can learn from our components. So, I'm going to dive into the third challenge, Dave. But and specifically for the lifecycle perspective, Goal 3, the technology innovation. So, any thoughts here, Dave, for a challenging perspective for technology innovation?

David Shepard: Yeah, so this is-- this, to me, speaks to some of the need to learn and absorb new technology. There is a number of tools out there, currently, that facilitate this kind of work. There's a couple of them that I would look at personally. One of them's called Open MBEE, which is a-- Open Model Based Electrical Engineering, I think, is what it stands for-- but basically, what it does is it has that central focus around the authoritative source of truth, which in their system is called the Model Management System, and then it has integrations for the various modeling tools to integrate with that, and you know, some extra tools for showing people how to build stuff and use stuff with that modeling environment. It's currently being used for what they call the 30mm telescope project. And it's mostly SysML models, you can check that out if you're interested in actually how they're doing it. But there's a lot of these tools out there. And you know, as always, it's an evolving space, so the tools aren't necessarily the primary focus here, but they are important to actually facilitating some of these things and allowing it all fit together and work.

Hasan Yasar: And so, I looked at the question window There is an interesting questions came from Elaine [ph?], and she wants to confirm that, "Are you using the same architectural models? And choices are difficult to change later without starting over." So, in the concept of rearranging, maybe you can open up a little bit that prospect your open architecture question. And the concept, Dave, as well. If you build up the models. If you can reuse that component over and over and again, but we are building more about the independent of an architecture, so each of its components can talk each others, right? So, the model, we build it, the concept, we build it. It's what is similar to the microservices concept in DevOps world. We build up the services, we build up microservices. Maybe we're using a container technology behind the microservices. So, each of the components that we are building in the pipeline, each of the components in the pipeline, each of the models, technically, can be open architecture, can be open architecture with that. We know how the model is interacting each other. We know the interfaces. So, without changing overall systems and just focusing on one of the capable change, that's going to bring about innovation, right? That's how the-- how we can deal with all this stuff.

David Shepard: Innovation and reuse.

Hasan Yasar: Right, innovation and use.

David Shepard: If you build your models around open interface specifications and focus on having that capability that that mod-- that that system models, self-encompassing and have the interfaces well-defined in-between, then you can potentially reuse those systems and combine them in ways that weren't originally envisioned when that component or module was developed.

Hasan Yasar: That remind a survey. Recently we had done the DevSecOps Survey 2020, what really we find out that if organization does more-- make sure it's in DevOps, their engineers have more time to learn. It was very interesting study that came out, and what they were telling me, if you really take out the so many steps from engineering world, if you take out there's so much better to send so much of the problem. Engineers are looking for more innovative new things. More improving the better components while you're learning actually, spending more time to learning. So, that group was spending more time if they have a DevOps, spending more time to learn. There was actually the percentage was very interesting, 80 percent of engineers are reusing the virtual platform to learn the new concepts so they can innovate themselves. Non-DevOps organization rate was 20 percent. It was a big jump from 20 percent to 80 percent, which is bringing innovation. That really goes back to some challenges here that we listed, and these engineering activities are disjointed across lifecycle, because we're not really sharing that. It's all disconnected because each of the model development, it's each of the model creations it's different lifecycles. So, we have to do the same lifecycle, same environment, same tools that we're using for the system that we are building for that A or that B or System B, System A. Or whatever product we are building the same lifecycle, we can use that. So, it's basically creating some innovation time for the engineers if there is more automations, more infrastructures, more of the common sources they're learning. It's not just they're learning from environment, it's the same they're learning from the mistakes as well. And I see a lot of efforts on the DevOps perspective. So, let's go look at Goal 4, David, about the IP Security Protection Infrastructure Methods. And there's some challenges here to implement Goal 4. Any thoughts here?

David Shepard: Yeah, for sure. You know, maintaining intellectual property agreements and managing digital assets that are easily transferrable across networks, it does present its own challenge to secure those artifacts and make sure that you're protecting those creations. You know, software has the same problem. It's not a new problem, it's just there are different concerns when you're building an integrated system of system sort of thing where you maybe don't own all the IP, or there are severe penalties for losing or exposing the digital model. And you could end up giving your competitors an advantage if they're able to get ahold those. So, yeah, there's not a lot of-- there's a lot of questions here as to the right tools to protect that. You know, the current thinking is I think that we are applying best practices for security to a lot of these things, making sure roles and access controls are in place and limit encryption everywhere of course and just making sure that those assets are as well protected as can be.

Hasan Yasar: It is very similar concept of building up a secure DevOps infrastructures, right? We are building each of the containers as an example, how we can harden the containers, each of the modules that we're using. And then same as how are we going to protect our infrastructure, which is our pipeline for the system DevOp pipeline that we are building the models, we are building components, so that we can make sure that they are secure enough. Like implementing the best processes in terms of security perspective. Using your own as control as an example. Like, "Who's touching my repository management? Or who's touching my build steps? Who are the people that configure in the build environments? What is my dependencies that I'm using?" which is build of metrics as an example. We heard about a lot of things in this topic, too. So, "How are we going to make sure that we are using it, it's kind of secure? We will have to vet it. We verify that components as part of our infrastructures." So, it's very related to this component like we're going to build up and turn solutions, which is the same pipeline, but also connected to the right infrastructures. We'll support the right tools, methods and process.

David Shepard: Yeah.

Hasan Yasar: So, we're going to have a common process that we're going to follow. So, there is a question-- yes, Dave.

David Shepard: Yeah, so the security problem is substantial and it's significant. But I think it's healthy that it's being talked about and people are thinking about it as part of their process of going out and building systems that support these sort of activities. Because I'm not entirely sure that that was always in the thought process, you know, from Day 1 is how to secure your intellectual property. Maybe not for the whole system anyway. So, I look at it as a challenge, but as a healthy sign that it's being engaged and people are thinking about it early on.

Hasan Yasar: So, let's look at the system requirements and think about the security at the beginning, understanding the risks, understanding the model or systems behavior. Try to insert security at the beginning into the lifecycle. That's what we have been saying for DevSecOps for years. Now the same concept is true for Digital Engineering perspective, too. But at the same time, you said the key point, too. Like the methods is also important, what I call security methods that we're going to use in infrastructures, all the techniques that we are going to use. Which is the overall pipeline, overall components. So, I'm going to jump to the training pieces. Before that there is a question on the chat room talking about the-- that came from Hubert [ph?], "There's quite a lot of similarities between Digital Engineering and Enterprise Architectures, and how does this framework contrast against something like TOGAF?" It's a good question, Hubert. And when we look at Digital Engineering, we are talking about digital representations. In the TOGAF end, which is open architecture framework that we are talking about specifically, so this is more about the organization or the system that we are dealing with, it's a common enterprise architect. But how we are representing that architect, Digital Engineering comes in the picture, make a digital representation of that model and thinking and carry out through the environment, so TOGAF is going to deal with high level and in the architect level, enterprise level, but the Digital Engineering, it's the capabilities that we are building and more modular-driven, more simulated environment, so we can have a digital representation of the model, or the components that we are using it in the lifecycle. Kind of like building agility, building more capabilities for a lifecycle. So, for me TOGAF is more enterprise architect, still we can have an architect for the model. Maybe, let's say, if we have the product that we are building, let's maybe put that down here, if we have a product that we are building, that product has multiple components, right? System A, B, C, whatever the components are, so how those components are interreacting each others, and we didn't have architects basically. We have digital architect our systems. How the system is going to communicate, how are they going to be interacting each others, and what the boundaries are, and then another question came from the comment as well in terms of what are the liabilities. So, we can define how are we going to build up this Model A in a part of our systems? Then we're going to use the Digital Engineering concept of make the model, and try to have additional representation of this model before building the hardware. Let's have the virtual hardware environment first, and then make sure that this environment is really matching our criteria, so we can simulate that and then we can build up the actual hardware module after that. So, I'm sure, David, you have a couple ideas about this Digital Engineering concept. Maybe you can open up to this, and how we are creating some virtualizations and other terminologies which is about the Digital Twins. Any thoughts here? Maybe you can open up a thought?

David Shepard: So, Digital Twin is taking the next logical step from just having a digital model as a shared source of knowledge of the thing that everybody is trying to build and integrate together, and actually creating a fully digital representation that you can run through a number of test scenarios and actually see what your system might operate like in a real environment if you realize that thing. Specifically, the Digital Twin as a definition would be you have a thing that exists instantiated in the real world, whether it's a jet plane, or a rocket ship, or whatever you're building. And the Digital Twin gives you ideally an exact replica in software for simulation and testing purposes. The thought that comes to mind on this one is autonomous vehicles. So, autonomous vehicles, self-driving cars, they're using these sorts of Digital Twin models to have a model that they can test their self-driving systems and actually drive them on like a simulated track, in addition to the driving that they have to do on real world streets and closed tracks for their testing. This is necessary in their case for a lot of reasons. Not the least of which is actually that they need tremendous amounts of hours, of actual testing of their systems to ensure that they're reliable and safe for people who are eventually going to be driving or riding in these vehicles, you know, validating that the systems are resilient and safe and secure. So, they create these Digital Twin simulations, so they can actually augment their testing, and then they periodically have to go through a process where they validate that the simulator is actually performing like a real car would so that they can validate that their testing is accurate.

Hasan Yasar: So, let's try to make a little sketch of what you described, David. And having digital models, it's the similar things in our software where we're dealing with a test harness. So, based on the things that we are building based on the feature or a capability that we are building a feature or a capability that we are building create that model as digital representation in a DevOps world, that digital representation of these events can be part of the continuous integration. So, based on each of the dev code changes and have the hardware virtualization environment ready, so we can put the new version of the code into the Digital Twins environment, see how this is boing go to behave, right? So, that can be initiated automatically from a CI server so we can build up an orchestration perspective, open up that environment build up in the layers and push the new code and test it out, and collect the feedback and then also share the feedback with other stakeholders, each of the stakeholder are responsible for the various phases of the lifecycle. So, maybe open up the Digital Twins concept here, David. We want to go a little bit deeper, we will talk about our infrastructure provision and concept, right? We're not just looking for a Digital Twins, it's just going to run by itself. We don't want to have digital representations running standalone by itself. Try to integrate it into the lifecycle, because we can collect the data. Do you have any thoughts on how can we-- any tools in techniques that we can use as the running Digital Twins? Like using infrastructure, provisioning concept or some other challenge?

David Shepard: Absolutely. So, there's a lot there at the technical level to actually making this work, right? You don't necessarily want your models running all the time. They possibly take up a lot of system resources, fairly expensive to run them all the time. So, you want those models to be able to be something that you can spin up and run a battery of testing on and then collect your data and then shut that thing down. And we see a corollary here to the way we do things in DevOps with our automated system testing and our test harnesses and provisioning infrastructures code, actually running those simulations. The difference here with Digital Engineering, if you're building Digital Twins of hardware models, you need different tools and different types of simulators that you have to instantiate and bring up in that environment to do that testing. So, whereas in software, we typically have a standard architecture in which everything runs today. That's not the case when you start working on some of these Digital Engineering systems. So, that the amount of hardware simulation, and the amount of physical and you know, you can even be talking about physics simulations and things like that that tie back into your models. So, there's some differences there with the software, but the enabling system that underlies those simulators is very similar concepts to the way you provision things, conduct your test, collect your data and tear things down.

Hasan Yasar: There's a process associated as you said, there will be a cost saving, if you don't need it, just tear down and use it again, because you can automate the process. So, it's going to save some time as scalabilities. So, I'm just going to dive into the fifth goal about the workforce. Let's see, there's another challenges about building up a skilled people and having the right metrics. So, any thoughts on the skills and trainings and metrics and challenges that you know the Digital Engineering perspective? So, you know, metrics is a tool that you use to measure how effective you are, right? Are we making changes that are helpful to us, and if so, in what ways? And if not, in what ways? And how can we get better? You know, you can't really just arbitrarily make the claim that doing Digital Engineering or doing DevSecOps is going to make you better faster if you're not collecting that data that shows you that you're actually getting better and developing capability faster. So, that's part of it, and the cultural aspect of the people who are engaging in and b building and working within these process models, you know, there's a lot of new technologies, a lot of new ways of doing things that you need your development organization to actually have a shared view of that and understand how to use all of those tools effectively. So, there's a significant amount of workforce training and stuff like that that goes into this to make it all actually work.

Hasan Yasar: That remind me DevOps perspective here, Dave. In DevOps, and it's a similar mentality we are looking for, how the people is going to learn what the skills sets is required. There is no way we can really get everybody the same room, train hundreds of developers, or engineers. It's not practical. So, again,, what I've been seeing more things about more learning when we need it, and then look at the system that we are building, learn at the moment. What about like an on-time learning or on-demand learning as the new way of building some skills, building up some learning concept? But specific for the security perspective, if I can open up a little bit here, we don't want to train everybody for the low-level security coding, which they're not really building up at that language specifically. Let's say if they using CC++, do they have to learn how they're going to fix the C++ powerful overflow vulnerabilities? Or if they are really writing a hard core C code, we don't really want to give them just a single injection, as an example. So, kind of giving the related training is very crucial. Same thing for the language perspective. There are many languages available. There are many tools available right now. But the key point is let's train the folks when they need information, when they need to learn something. That's another thing that came from our DevOps survey, people are learning, especially for the things by spending time looking some virtual platform or other component when they needed it, when there is a need. Specifically, when there is a need, they even look at learning pieces and specifically look at that ideas what needs to be covered. So, that's kind of more practical perspective of learning based on what they need that they have. And you brought up the metrics question that goes back to our DevOps automation. If we had-- maybe let's go a little bit this side, "If we have a full end-to-end lifecycle we are talking about, end-to-end lifecycle, so we can collect what type of metrics in our pipeline, so typically metrics getting managed organization. Define what metrics that you have, and then visualize these metrics with organization you can learn." So, kind of related together. So, we can really increase our knowledge pieces by collecting right metrics. We can measure it, and we can get better out of it. So, changing more incentives, understanding how things we are building and looking for each of the components if the metrics measurement and including our capabilities, we can learn better overall perspectives. So, we have a short timeframe that I'm going to jump to the next topic, Dave. Let's take a look at what is next. Let's go back to our agenda items. Where can we start? And then it's about what is next, where to start. So, we learn a lot of things about Digital Engineering. We looked at the DevOps. Where do we start for implementation if we are in a new environment, we would like to go that route, maybe start from DevOps or Digital Engineering. Or what is your thoughts to starting and for implementing any new concept?

David Shepard: Yeah, so I always like to start with requirements. You know, what is the thing we're building? And how can I use the process model and the technologies to help me do this next new thing better? And start to realize some of these goals. It all starts from those requirements. There's so ma-- like you said, there's so many tools out there that people can use. By really starting with a focus on what you're trying to build and where it's going to get you to as your first step, it'll help guide you on that process to actually-- to make some of these decisions early-on, and to try them out. And you know, that's also another aspect of this is you might not make the right choice the first time you choose a new tool. It might seem like a great idea, and then you get a little ways along with it and realize, "This just isn't working out." You know, that learning experience, especially early on can be very valuable to help you from going down a path too far, you know? Learn from the mistakes, fail fast and move on to something else that actually works.

Hasan Yasar: So, if I hear you correctly, we would like to start from the "Why?" first. What is our goals? What we would like to achieve? What is our requirements? What needs to be achieved that? So, ask yourself as an organization, "We need a DevOps, fine. What is the problem that we would like to solve for specifically for DevOps implementation? Or what is the reason you're going to go Digital Engineering?" Understand the benefits of Digital Engineering, and maybe that's going to be your "Why?" There is actually good strategy documents, and as has been shared from the 2018 Digital Engineering Strategy, there are many benefits. So, we can look at the different benefits. Maybe these other benefits can be your "Why?" Why you need to start that, and look at that, understand. So, if you know the concept, if you know what needs to be done for a specific perspective, then you go back to the question about yourself in terms of, "What needs to be done. What type of actions we have to do to achieve that?" Like our goal is to have-- build up a source of truth, as an example. Maybe can build up repository management, that's our action, that's our task. We're going to build up a repository management. Then we look at the how question is, "Okay, you may be using a GitLab, maybe using other type of tools, and see which one is available for you." Ideally, you can relate, bend the tools, but don't bend your process to make the tools that will be a part of your case. We will bend the tools to achieve your process, to achieve your what and how that you would like to do.

If you're really looking for the tool-oriented implementation, and some of you asked the question about referenced architectures, so there is reference architecture with respect to DevOps and in our SEI website, you can go download it as well. But more specifically, try to really bend the tools based on what you needed. Don't really let the tools going to drive your capabilities. Because otherwise, it's going to be a problem, it's going to be a lockdown [ph?] situation, where it will be very difficult to go back and change anyone. So, as that's said, we have about a couple minutes to go. Any lost minute, any last type of word from you, David? Any takeaway that you would like to mention as the last couple sentences from your end?

David Shepard: You know, I would say that anything that you can do that is going to take you down this path of quicker more iterative delivery of capability and learning is going to get you closer to your goals that we've talked about throughout. You know, it's about-- there's a reason people are spending their time learning and trying new things because they really need to. They need to spend that time learning, because the world is changing very fast. And this is-- this sort of thinking is how we deal with and keep up with that change.

Hasan Yasar: Thank you so much, David. Since we have been out today as a last word, I would like to say little learn, fail fast, because these learnings will tell us it's much easier to fail. If we're not going to be failing fast in a hardware environment, so we can learn fast as well. That said, I'm going to hand it over to Shane. Thank you so much for attending our webinar today and looking forward to next webinar as well.

Shane McGraw: David, Hasan, great discussion today, and again, think you for sharing your expertise. We really appreciate it. And lastly, we'd like to thank you all for attending today. Upon exiting, please hit the like button below the video window and share the archive if you found value. Also, you can subscribe to the SEI YouTube channel by clicking the SEI Seal in the lower right-hand corner of the video window. Lastly, join us for our next live stream, which will be March 24th, and the Topic will be "Pivoting Research in Quantum Computing Field, Responding to the QAOA Superpositions" by Jason Larkin and Daniel Justice. Registration information is available on the SEI website now. It will be emailed out as well. And lastly, any questions from today's event, please send to info@SEI.CMU.edu. Thanks, everyone! Have a great day!