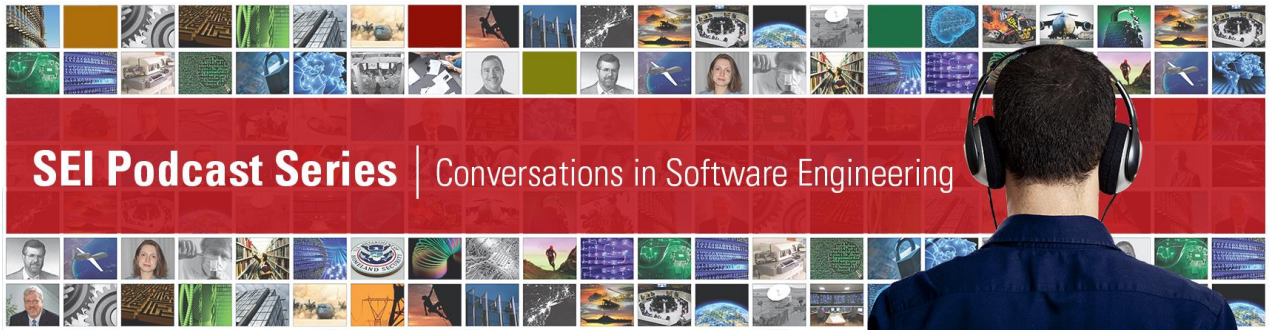

Agile DevOps

featuring Hasan Yasar and Eileen Wrubel

Eileen Wrubel: Welcome to the SEI Podcast Series, a production of Carnegie Mellon University's Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the Department of Defense. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

My name is Eileen Wrubel, and I am the technical lead for the SEI's Agile-in-Government team. With me today is Hasan Yasar, who is the technical manager of the Secure Lifecycle Solutions team in the SEI's CERT Division. Hasan and his team also regularly publish on the SEI's DevOps blog.

Earlier this year, I spoke with Hasan about Secure DevOps. Today, we are going to do something a little bit different. Hasan and I are going to talk to you today about Agile and DevOps and how they can be employed together to meet an organization's needs.

Hasan, would you do our viewers a favor? Tell them a little bit about you and your work?

Hasan Yasar: Thanks for the introduction, Eileen. As you described, my team and I do DevOps work within the SEI as part of CERT. My team frequently assesses the DoD site specifically and looks for their capabilities and how to implement DevOps and how they can include security in the software development process. We are also looking for industrial best practices and bringing that to the DoD specifically.

Eileen, we do so many other things as well. It [the DoD environment] is a totally different environment than in industry, and we compare that [the industry environment] to DoD environments, maybe because of highly regulated environments. There are many other restrictions, many other policies, and security controls, which are not applicable in the industrial domain. DevOps, then, is more about the federal [environment]. When you try to get more into DoD, it is really changing the concepts. So, my team is going to help the DoD, and really help with implementing DevOps principles and their process. That is the typical work we do in SEI. You are doing that kind of work as well, so it is complementary.

Eileen: It is. We focus primarily on helping with Department of Defense and government program offices who acquire software from commercial or from defense or government laboratories. We help the folks who have to interact with the software engineering team when that team is using Agile processes and Agile methods.

As you know, you can break the process if you do not have contracts and other business structures that are aligned with the software development and delivery models. So, us working on the business end and the interface to the technical teams combined with the work that you are doing with the DevOps organizations, I think that there is a really good marriage there where we can apply the principles together and achieve better benefits for our customer programs.

Hasan: That is great because when I work with our clients, I see the constant requests about the process-related [items], which is where Agile comes into the picture, and talk with the acquisition team program managers' offices, which is one of the enablers in the DevOps mindset. I am just going to go over the background of how DevOps started, because we should understand where it comes from so we know [how] we can talk about the Agile mindset.

Originally, the DevOps terminology started in 2008 in a conference. That conference was an Agile conference, an Agile infrastructure and operational conference. So, Agile people—not that I am blaming you—but Agile teams were producing a lot of code because the principles, and then the manifesto, says just constant requests and constant changes. That actually resulted in a problem on the operational side.

So, a person named Patrick DeBois, he was kind of thinking, *How we should do an operational mindset to ingest, to get a lot of code changes?* So he said, *I am going to present Agile infrastructures in Agile conferences*. So, changing infrastructure mindset to support Agile momentum. He did not show [this?] in his conference [because] it was, kind of, funny things because nobody was believing that the Agile concept [fit] in operational world.

The following year in 2009, John Allspaw [and Paul Hammond] presented *10+ Deploys Per Day: Dev and Ops Cooperation at Flickr*. This is the first time DevOps came into being, history wise and in the [development] communities.

So, in 2009 Patrick DeBois opened up DevOps days in Belgium, and he used DevOps Days as a term over there. So, DevOps term came up from 2009. When we look at the history, it is basically a reaction to the Agile team, and *How can we get the Agile work into the DevOps mindset*? That is how it starts. That means that, literally, changes require some operational mindset changes. There is no standard definition of DevOps because it is so much of an emerging term, and everyone has his or her own definition.

If I say my own definition as part of the SEI, maybe myself and my team, DevOps is basically a set of practices, including the communication, the collaboration, and also the constant communication effectively [among] all stakeholders, not only *Dev* and an *Ops*, but all stakeholders, including program managers, including the acquisition folk (as necessary), and including the security experts, which most of the time are often forgotten. Get all these folks together in the same visible environments and, using the similar platform or using similar toolsets, let them communicate with each other. So, that is my broad definition of DevOps.

Eileen: So, the timeline here is interesting because the 2009 timeframe is about when the DoD came to the SEI and said, *Hey we have heard about these Agile approaches.* They asked us to really look at the technical feasibility of employing them in the complex, regulatory, statutory, and policy framework of complex weapons systems and complex defense programs. As we started looking into, *Does this violate any of the policy? Does this violate any of the principles of the federal acquisitions system?* And as we started to see more and more programs taking off, you saw DevOps growing and maturing. We have come to this serendipitous point where the use of Agile is really maturing in federal and defense programs and the practices and the tool sets in the DevOps world are really, really solid. We are at this fortunate confluence, if you will. Just as you talked about DevOps coming back to practices and principles, we get asked a lot on the Agile team, *Just give me a checklist. I need a checklist to see if it is Agile.* We take it back to principle.

Hasan: It is a good point, actually. Nobody will say, *Is it process to use these tools and these techniques?* because that is not the concept of Agile. I am sure you are the expert—it is more about the policies, more about the process. The mindset of DevOps is utilizing the best practices, utilizing the tool stack, and having it transfer [and] basically be an enabler for the Agile team.

I hear many stories when I do the DevOps assessments. People know how to use Agile techniques or the process. They maybe do the Daily Scrum or maybe do the Extreme Programming, or maybe the Kanban. However, knowing the techniques is one requirement, like a people-process side. But, if they do not have any tooling section of it—let us say if they do not have any deployment pipeline—if there is no continuous integration; if there is no wiki page [or] anything like that, it is very difficult to communicate. What I see as a problem is, it is very difficult to communicate with all the stakeholders. DevOps comes in the picture and makes a good technology, a good process, and good techniques to help the Agile set of thinking, and this [is] part of the DevOps process. So that is kind of a good relationship.

When we go and do the DevOps work, regularly people have to know the process side of it as well. That is the relationship I see. We have done a lot of training together. We teach the people how to use Agile and then teach them how to use DevOps. That kind of goes hand-in-hand together, because if you drop one thing, there is no way to make the other thing as successful.

Having a great tool stack that you can set up. You may set up a perfect CI [continuous integration] server. You may set up a perfect issue-tracking system. You may have a great deployment pipeline. Maybe it is all integrated, but people are using it. People like the business analyst, or you just mentioned the program managers, and developers are using it. It is requiring some discipline, right? Disciplines like in the Agile world. What are the disciplines that have to be in place so people can take [advantage of] it?

Eileen: Right. We have to do things like change how we write contracts, so that we understand how the deliverables are going to be different. We have to understand that we are going to measure progress and quality of not only the technical product but of the processes building it. The way that we exercise that kind of oversight is very different from the way we used to do it. Having the tool sets and having the communications platform is all a really important part of that. Agile does not get you that by itself, but the tool stacks in your DevOps environment are going to help you go a long way in that regard.

Hasan: I am sure that I am going to ask this question, actually it just came in my mind. If anybody from the program manager prospective, if they say, *OK, Agile says, Let's get the stakeholder involvement at an earlier stage. So how are we going to get the user involvement at an earlier stage?* User means the end users. I am sure you heard that question at the other context or environment, or a maybe high-level environment, *How will they connect to the environments* or *How will the Agile team present wherever you are as of today or as of the end of the sprint?* Do you think you could come up with any questions like that they are asking to you?

Eileen: It is a tough nut to crack sometimes because it is all going to look different, depending on the operational context. If we are building a fighter jet, then that is a very different environment than building a business system. The way that we can interact with a financial analyst who uses the business system on a day-to-day basis, or a personnel officer who uses the system on a day to day basis, that is a much smoother chain of interaction, probably, than involving a fighter pilot who is climbing in and out of the cockpit every day and also getting that individual engaged with the development team on a regular basis.

We have worked with programs in a number of different ways to identify the people and the interaction mechanisms, and whether we involve surrogates from different organizations, what kinds of communications techniques we use. [There] is really a lot of evidence that it is not just a one-size-fits-all process.

Hasan: That is true. There is no blueprint. It's the same thing for DevOps as well. There is no way you can say, *Here is the tool stack,* or *Here is the best practice. Go take it and use it.* It is not like that. It depends on the organization. It depends on technology. It depends on people. It depends on whoever the stakeholder is.

My question is, really, when we get feedback—think about the bigger projects, direct from end users. They have to get feedback. So, they are giving feedback on what? The running application? Or, getting feedback on a piece of drawing paper? Or, getting feedback on what?

Eileen: How are we demonstrating it?

Hasan: How do we demonstrate Agile principles, that they get their feedback, right?

Eileen: Right, and getting them as close to some kind of demonstration of the code, whether that is in an integration lab or whether that is on the desktop in the environment that they are going to be operating in. We do that in a lot of different ways. Sometimes we use proxies for those people when the larger community is not available. Every six months you might see a user community come together to see the pile of releases that have successively evolved and see a large-scale demonstration. It is really cool to be at the Pentagon and have somebody watch a demonstration of something being deployed in real life.

Hasan: You said a key word here, a very key word. You are going to show something is *running*. Something is *working*. To show a user something is *running* and *working* is much better than saying, *I am going to work on it. Here is my plan. Here are my PowerPoint slides* versus *Here is the code. It is working. Here is my background and maybe the framework is going to work* and present something that makes more impact to the user. That is what I have seen [in] the DevOps work, because showing feedback like showing to the end user, who are the stakeholders, *See something is up and running.* It is giving trust. Sometimes, as a human, when we see something is up and running we may give feedback directly on the application itself versus reading documents.

That is kind of how DevOps is changing the world. It says, *Give early feedback*. How? It is using the technique, like the continuous delivery or continuous deployment, as it is going into production and the mechanisms behind all the continuous integration. If multiple teams are working together…. Think about the DoD context, there are systems of systems. There are many systems working together. If you have multiple continuous integration lines up and running, we will be able to integrate all the systems and then deliver them to an environment, probably a testing or staging environment, and get their feedback. So, Agile principles probably say, *Yes, let's get their feedback.*

An engagement, you describe an engagement as, *Here is how you do it in a DevOps mindset*. You decide it. You have a sprint planning—probably depends on the framework that is being used--maybe a two-week sprint, maybe a three-week sprint? At the end of the sprint, you are showing something is up and running. I saw great feedback because people may change it. Say,

OK, I do not like this. I do not like that, but I was meaning something else. Then I drove something, and I change my mind. Is that the Agile principles?

Eileen: Right. Then having everything riding on an infrastructure that lets us make those course corrections with minimal waste, [that's] really where I see all of this coming together.

Hasan: Actually, related to program managers, I think people are not really talking about it. In the Agile world, we are seeing that the program manager has to change their thinking. So, when they look for tasking on their work, how are they going to decide, *Are we done*? or *When are we going to get the feedback? How is the Agile the mindset thinking*? Can you talk about that?

Eileen: What we try to tell people is to focus on the high-level capability that we are trying to deliver. *Is it an airplane that flies so fast and goes so high and delivers certain kinds of munitions*? *What are the high-level objectives that we are trying to deliver*? *Ok, so we are fixed in that space. We cannot do things that violate those conditions.* When we get down to the implementation-level decision, we are trying to centralize the kinds of decisions that are long lasting and have significant economic impact and then decentralizing decisions that are frequently made and that rely on design and technical expertise.

One of the things that we are trying to work with a lot of our customer programs on is giving careful thought to a decentralized decision-making framework, if you will. *If you can make performance improvements to the tune of x, and it affects this other characteristic by less than y.*

Hasan: If I understand correctly, it is not a hardcore requirement, it is more about the capabilities, right?

Eileen: Right, about what the outcomes are that you want the system to achieve.

Hasan: So, changing mindset instead of having one-by-one requirements and, maybe, 1,000 pages. Putting in more about the capabilities, and capabilities is basically a feature. It is kind of like that is the mapping: get capabilities, map the features. Now, in the DevOps world, as a dev team, I will have to make sure that feature's in place. How? I am going to use my continuous integration servers and get all those features, build it, and show it to either the program managers or the users or the security folks.

Eileen: Then we know it is done, because we have agreed in advance on what it looks like when it is done. We do not ever get to the end and say, *Oh well, you brought me a rock, but that was the wrong rock*. We have already agreed on the performance parameters and the cybersecurity requirements and various other aspects of what it means to have successfully delivered that capability. So, when we demonstrate it and when we subsequently test it, we know what it means in advance to be done, and so we can be done.

Hasan: The *done* means for developers, *I finished up my job. I finished my functions. I delivered it.* But, actually, a done definition is different, right? An Agile term is done. What is the definition of *done* in Agile thinking?

Eileen: Right, that includes things like, *I have passed all of so many tests. I have got this test covered. I have not broken the build. I do not have any defects over such and such severity. I have described what the outcomes need to be, and I have verified those through the testing process that those outcomes have been achieved*. But it is much more than just about, *OK I coded these few lines and I am done*. It has to actually be successfully implemented and verified.

Hasan: This the DevOps mindset. *I finished it. It is working [in] my mission. That is it. I am done*. Actually, *done* is totally different because the thinking is that somebody has to approve it. Or, the definition of *done* has to be that the user has to use those features. If the user is not using it, I do not think it is going to be done. It is not concluded yet. So, even when you pass the testing, when you pass the acceptance testing, if it is not yet being used by the end users, it is not done. So, at the beginning of our discussion we said the definition of DevOps is really more broad because I want to make sure that, in my world, really getting the requirements and then delivering it to the production environment, it is up and running. When it's up and running, that means it is done. So, not only for dev people, because I see many organizations and people are doing Agile in a dev team, like having a daily scrum or sprint in a dev team, it is not a good practice, right? It is only limited to the dev team.

Eileen: Right. It is not just about the development team writing the code and washing their hands of it. This is why, with Agile approaches, we build in the demo, and we build in the retrospective, and we build in those constant feedback loops. If we did not get it right the first time, we can go back, and we only pay a minor time penalty. All of the DevOps infrastructure gives us the ability to be flexible about those changes.

Hasan: I guess one of the principles of continuous feedback is the Agile principles and Agile manifestos, correct?

Eileen: Right, and that sifts through a lot of the principles. We talk about welcoming changing requirements. We talk about business people and developers working together collaboratively throughout the project and satisfying the customer through continuous delivery of software.

Hasan: Continuous delivery is already part of Agile principles. In a DevOps world, we are seeing continuous delivery. We are seeing continuous integration. We are talking about continuous deployments. These are the key principles of DevOps. As techniques, these make all of these Agile principles achievable.

If you are going to get the business stakeholders involved, then a couple of techniques [are available]. I can, maybe, invite them into the meeting, or share documentation, or share the direct result with the end users or the stakeholders directly. *How you do that*? You are going to build the code, and then you can test it out, and then deploy it into the staging environment. So, we talk about the continuous integration as a build server having multiple teams together and working on it.

Then we will do the testing, which is based on the acceptance testing and maybe part of the requirements gathering. Then, after the delivery into the staging environment, the user can see and [provide] the feedback. So, the quickest feedback is achievable. It is really important in terms of the software development lifecycle. If there is a feedback six months later, if something is wrong, what is the cost of that?

Eileen: Right. I do not remember what I did six months ago. Do you remember what you did six months ago?

Hasan: I do not remember, either. The context switches and also the effort. If something is wrong—in terms of a decision, in terms of a feature set—you may lose six months of effort. Nobody is really thinking in that mindset. How much more quickly you get the feedback, and you can save a day maybe, if you get [daily] feedback. If you build it at night, and then tomorrow the team will see the results, this is possible. Having continuous, continuous deployment, we can get early feedback directly from the user a day after and [you will lose] only one day of team effort. You do the basic math: one day of team effort for five people versus a six months of effort for a team of five people. Maybe a waste, so there is a lot of eliminated waste when using Agile and DevOps principles together.

Eileen: We all know the truism about correcting any kind of defect: The further away you get from the injection point, the longer it takes to fix and the more expensive it is. So we are basically just squishing all of that to left. Even if it is a defect in how we have articulated in what we are looking for, having those rapid feedback cycles, by definition, is going to save us. It is going to save us that effort. It is going to buy us economies in terms of time and cost.

Hasan: In the DevOps mindset, there is an idea that is more about the continuous of *everything*. That is the lead concept that is evolving, because they have continuous integration and deliver everything else, and continuous of everything.

Everything means whatever you do in your software development lifecycle. Make it a continuous evolvement: continuous and incremental changes or the continuous testing. It means everything, basically, including security as well. Do you see any problems in the security communities that

do not like the Agile thinking? What is your take on this so far? Do you see any pushback from the security folks?

Eileen: Historically, at least in the DoD environment, a lot of the cybersecurity aspects were dealt with in a compliance-oriented fashion. So, an independent agency is taking a compliance checklist, if you will, and applying that to a software system at the very end of the development lifecycle and shifting the idea of cybersecurity and testing to a continuous lifecycle activity. Bringing in the risk management framework, and really building those activities in throughout the lifecycle, has been a challenge when you get pockets of organizations that are still trying to apply those in very compliance-oriented ways. Taking great consideration to maintain their independence, because they are required to be an independent organization. It can't just be the software engineering shop doing its own compliance audit, if you will.

So, getting away from that compliance-audit mindset to one where we are building in the verification and the validation for cybersecurity incrementally as we go, building that into the automated testing, building that into the continuous integration. Getting to the point where we are achieving that automated governance, if you will. We are making steps in that direction, but we are definitely not there yet.

Hasan: It is really still a lot of things. There is work to be done conceptually. Agile says, *Change*. For any work is a constant change. But the security folks, if they know what the change means, they can prep themselves. You describe having a continuous testing, or the security testing, or including any security controllers up front. Once we share that controllers are in place, which are coming to the pipeline, the security folks see, *Here are controllers in place*. They will see they are implemented. Then they can verify by testing. So, basically, taking a change, making it visible through the DevOps pipeline. Because an Agile person is going to say, *Yes, I have to include security into my backlogs, maybe not backlogs actually, my sprint or my epics or my features*…

Eileen: My definition of done, my acceptance criteria.

Hasan: Yes, acceptance criteria is a good point. *Here are my acceptance criteria. Here is my boost-case testing probably*. Right? *And then who is going to tell the boost-case testing, by the way*? The security folks should tell them. Whoever is managing the team is a scrum master, and they can call the security folks. *Hey, tell me the acceptance criteria for those features*. Then the team, as a DevOps team, they can take those features and test and write test automation into the continuous integration server. Then they build it and test it and verify it is working. It is, basically, a big gain for the security folks. They are not really aware of it so much, but [take the] types of tools they were using over in the security testing, and bring them to the DevOps world, bring them to the continuous integration, and let them do the continuous integration testing.

Ideally, we are not there yet. If we give test features all the way back to developers, they will test it right away without writing a code. Then carryover the Agile team as the scrum master. I am talking about scrum as of the framework, right? They know they can take the concept of testing and security mindset that they are talking [about] in their daily scrum as a team. *Hey team, we are writing authentication features in our application. Here is a security testing, and we should know about it*. So, it will tell and then the team will go and write test automations and test that and verify that. That should be really a feature of the other software development that I see so far.

Eileen: We are going to have more chance to talk more about that with Agile and DevOps and the idea of continuous RMF (Risk Management Framework) in a podcast that we are going to be recording here shortly.

Hasan: We will have a next series about that.

Eileen: Yes, so we will put in a plug for our next podcast while we are talking here.

Hasan: Absolutely. I am going to talk about [that] myself, as well. What is your next SEI [project]?

Eileen: We are working really hard to figure out how we can continue to extend—if we take this again back to the principles—the acquisition of software-intensive systems in the DoD is a long-term proposition. We continue to operate in constrained fiscal environments. The operating environment changes around us. We continually need to deploy new capabilities, high-quality capabilities to our end users more rapidly.

What we are trying to do is look at, really, that whole acquisition lifecycle, and how we can apply these principles of iterative-and-incremental approaches to small batches that value flexibility. How do we build that into various processes throughout the lifecycle, not just the software engineering? The software engineering is really only one piece of what it takes to successfully deploy a software-reliant system. That is really what we are trying to focus on, is how broadly across that frontier can we apply those principles?

Hasan: DevOps is going to be here to help you. As a segue to what I do, I am planning an SEI DevOps team to carry out the DevOps thinking to the architectural level as early as possible. Because the principle says loosely coupled architecture. If there is no flexible architecture in place, we cannot make it Agile. We cannot make it DevOps, because automation and testing require architecture changes. We are going more towards the architectural changes, and then how does that affect the relationship with DevOps? How can we carry out the small, small pieces and make it a bigger piece based on architecture, based on deployments, and then build the environment? So it has to be fully DevOps from beginning to end with the help of Agile.

And, a little more training as well. I believe we are going to do collective training as well. Because, as I said, telling the people tool-side, and practices [is one part], but they have to know the policy side. They have to know how to do it and how to use it. It is requiring some discipline.

Eileen: One thing that we see a lot, that I will put in a plug for, we see people say, *OK, DevOps. That is the thing we do at the end. We are getting ready to figure out the intersection to getting things deployed. Let us bring in the DevOps team at the end*. That is exactly the opposite of what we are trying to do. We are trying to start at the very beginning and say, *Well, what does continuous integration mean here? How can we benefit from this even before we have ever written a line of code, before we have completed any concepts on the architecture?*

Hasan: Once we build up the DevOps pipeline, it is only a one-time job. Then, building on top of the features applications, once we have the good deployment pipeline, we can add all other features that are application-related with an Agile mindset. I describe DevOps as building a highway. Once you build up the highway, we have people riding a car on it with Agile thinking.

Eileen: There we go. They are driving the car fast enough to get there, but not so fast that if they need to make a right hand turn they lose control of the car. That is really what we are getting at.

Hasan: Checks and balances are in place on that road, so that we can really monitor it.

Eileen: Great. Hasan, thank you so much for sitting down with me today. This is a lot of fun for me. I cannot wait until our next talk on Agile and DevOps and continuous Risk Management Framework.

Hasan: It was a great discussion with you. I really enjoyed it. Thank you so much.

Eileen: Thanks. Hasan and his team regularly author posts on the SEI DevOps blog at insights.sei.cmu.edu/devops. Please know that we will provide links to all of the resources mentioned in today's podcast in the transcript.

This podcast is available on the SEI website at sei.cmu.edu/podcasts as well as Carnegie Mellon University's iTunes U site and the SEI's YouTube channel. As always if you have any questions please feel free to contact us at info@sei.cmu.edu. Thanks.