How Risk Management Fits into Agile & DevOps in Government

a Roundtable Discussion with Eileen Wrubel featuring Tim Chick, Will Hayes, & Hasan Yasar

---

Eileen Wrubel: Welcome to the SEI Podcast Series, a production of Carnegie Mellon University Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University. 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 program. Today I am pleased to welcome you to a roundtable discussion about the Risk Management Framework and how it works with Agile and DevOps.

I am going to have my colleagues introduce themselves and tell a little bit about their role at the SEI and their work, and then we will delve into our subject matter for the day.

Tim Chick: My name is Tim Chick. I work for CERT's Security Automation Systems [as] technical manager. Our focus on my team is cybersecurity, because it is part of CERT obviously, but it is also focused on systems engineering best practices, software assurance, the Risk Management Framework, and how to build and sustain secure systems.

Hasan Yasar: I am Hasan Yasar, technical manager of CERT's Secure Lifecycle Solutions group under the Security Automation directorate, which is in the CERT division. My team does engineering practices and product development based on SEI clients. While we are doing engineering work, we are improving and also increasing DevOps awareness, which is addressing DevOps principles and processes.

Also, we are addressing the security needs of application requirements from beginning to end. That is our main goals and main practices.

Will Hayes: I am Will Hayes. I am the principal engineer on the Agile-in-Government team here. My work is primarily with Department of Defense programs that are implementing Agile capabilities within the context of a DoD-5000 framework. So we are helping acquirers as well as development organizations understand how to use Agile concepts in that setting.

Eileen: Since we have audience members who may not be familiar with all the terminology, can the three of you briefly give me a summary of what it means to talk about *Agile* and *DevOps* and *RMF*. I'm going to start with Will.

Will: With Agile is the notion of incrementally and intuitively building a system, starting with some prioritization process by which we take the hardest part, or the most valuable part, or some way of differentiating some small piece that should go first, and quickly iterating and learning as that work happens so that subsequent development activities are informed by the outcomes of the early iterations.

Hasan: That is a good start. I am going to say DevOps is the extension of Agile. DevOps is a set of practices and principles that increase the communication and collaboration, mainly of the IT/operational folks. As we know, the software development practice is requiring all other stakeholders as part of the development process, and mainly security folks. So, DevOps is basically addressing all this communication and collaboration and also increasing automation amongst the team members. We can implement more Agile thinking, more Agile process throughout the lifecycle.

Eileen: OK, great. Tim?

Tim: RMF stands for risk management framework. It is a NIST publication. Basically it is the way in which the government, different programs and agencies, get their authority to operate. With any system, when a program manager does trade between cost, schedule, quality, and functionality, they are inheriting risk. Someone needs to accept that risk before it is transitioned into operations, especially in the Department of Defense and the larger agencies. This authority-to-operate [ATO] construct is where an authorizing official accepts that risk especially from a cybersecurity information assurance perspective. RMF is the current process in which those authority-to-operates are granted.

Eileen: A lot of our DoD-client programs and collaborators have a historic experience with that certification-and-accreditation process where every three years you bring a compliance checklist to the certifying official. At that point, you are granted your authority to operate. They ask us frequently, *How do I jibe the new RMF process with what we are trying to do to innovate with Agile and DevOps where we are relying on small batches and regular, frequent releases?* If you can, start us off talking a little bit about how RMF fits into that world.

Tim: Traditionally, government uses what it is called DIACAP. You can Google it. It's Department of Defense Information Assurance Certification and Accreditation Process. I think it is accreditation and certification. The idea there is very much compliance-based. With the new NIST standard, it's now *risk* as well as *compliance* come together, *How do I make that assessment?* Because of that, the tradition of every-three-years ATO and the annual reviews, that part has come with it.

But RMF—if you read the material and you look at it—there is a construct built into it already, which is a continuous reauthorization construct where the AO or authorizing official basically says, *Based on that initial assessment, I believe you have enough infrastructure in place to maintain the current set of risk that you have accepted at a minimally acceptable level, even though you are continuously making changes to the system*. But that requires a lot of trust between the authorizing official as well as the program offices themselves in terms of—that they have the right contract, they have the right things in place.

That is really where the government is currently struggling is, *What is enough? Where is enough infrastructure in place that the person assuming the risk, the authorizing official, they trust the program office?* That is really at the crux of the issue. That is necessary to become more Agile, to be able to get to this lifecycle. Really, the only way to do that is through some automation, is through the DevOps constructs.

Hasan: To look at what Tim said, I think we need to understand why the problem is important. First of all, security is based on risk. If you know what type of risk you are dealing with, then you can talk about the security.

The other thing is if you are giving the developer some talk about the developers' perspective. So developers, they don't know, really, what they need to look at, *What are the requirements related to risk and related to security?* Because there are many controls available, which one do they have to address? As Tim said, *When is enough enough?* How are they going to do from the developers' perspective that I have to look at it? And I share the risks amongst them.

The other thing, also, [is] that DevOps will help with mindsets, which is basically sharing all the information through the secret officials, which is creating a trusted person [who is] not the program managers is required. Also, a developer should write the code. Then, if security is part of the code, which is giving it visibility to either the program manager offices or those information security officials, which is build it in the system. If it is a manual process, there is no benefit basically. You can go and check it out.

The main thing is, we have their automation. That is something the government is struggling with right now. Not only governments, almost any organization is required to be compliant for compliances like HIPAA, SOX [Sarbanes-Oxley Act], or RMF in the DoD context…..

They are struggling because it is a manual process. Somebody has to approve it personally. Somebody has to say, *Yes, I assure that this system has the security controls on it and has passed the test. I can assure that, so I can release it into production.*

Now, with the speed of Agile, with the speed of the DevOps mindset, *How can we connect the requirements and then make it efficiently, so we can release quickly?*

Tim: Down that same path, traditionally DIACAP very much was an afterthought. *I built the product, and then you come in and accredit it, and you determine whether it is an issue.*

RMF is very much designed to be part of the entire software development lifecycle, the entire acquisition lifecycle. At the very beginning, before you even begin doing requirements, you are supposed to think about the categorization of your system so you can figure out, *What are the requirements?* That corresponds to various levels of controls, which are things like, *You need to use multi-factor authentication* or, *You need to have specific types of configuration management techniques in place to ensure that your system only does what it is supposed to do and nothing else.*

Will: I think what you are describing describes a larger trend we are seeing in Agile implementations of shifting things to the left. Whereas things that provide us a level of assurance have traditionally been viewed as something we do when everything else is done. That way, we can have the full picture. I think with the success we are seeing in Agile development, we are seeing an ability to get that picture without everything being done and incrementally improve the amount of information that goes into such decisions. Certainly, tools are a huge part of that.

RMF, I think, adds another flavor to this in that there is a forward-looking attitude to have a notion of where the risks are and maybe have that drive some of your choices in development. I think that is a really exciting frontier for us.

Hasan: The one thing I really like about the RMF concept, it is a continuous process actually. There is a six-step continuous process. When you put RMF aside and you look at DevOps and Agile, we are looking at a continuous process as well. So something is

moving along. We have the incremental release, continuous integration, continuous deployment, and continuous delivery. It's basically a lot about *continuous.* The train is moving through the pipeline as an application. Now RMF is basically another continuous process. There are six steps. So, if you get that map, the continuous application lifecycle, that is the integration piece. One important piece we should be looking at is, *How can we integrate that process into the application lifecycle*? We can increase the automation, so we can add more security checks on it.

Tim: With security checks, it used to be manual because there were no tools out there. Now there are tools out there in the software assurance realm that really can evaluate my software and determine, *Am I using good coding standards? Am I writing code that makes my code more vulnerable to hackers or malicious actors?*

Most people don't write all their software. Most of your software is a collection of libraries or other applications packaged together to solve a unique problem. When you start taking in other people's products [the question becomes], *How do I ensure the integrity of that supply chain?*

Well, there are tools out there now that can say, *You are using that open-source product. Well, this version has a known, reported vulnerability in it. You shouldn't be using that version anymore.* It can automatically evaluate your software and provide those developers the type of feedback throughout their development lifecycle and not wait until the developers finish. They have totally used that package to get everything working. Then, at the end, under the traditional approach, a security person would come in and say, *You can't use that version. Now I have to redo all this work*. I'm going to have to get reauthorization. *I also have to retest everything from the functionality perspective, too, because I just switched up my library package*.

So, *Give me that feedback as I develop and I can make these changes when they're cheap.* I mean, it is basic quality stuff.

Will: That cost is a really big issue because most people when they think about these quality-assuring activities, they think of the cost. They think of the experience they have had when they are disrupted because they occur at the end, when costs are steeper. Now, with a forward-looking approach, a continuous notion, we have the authority to invest in assuring quality when it is most beneficial.

Hasan: I don't think we have the luxury to wait six months to reassess our system, because adversaries are using the similar sources our developers are using. It is open source. It keeps growing a lot.

If developers are using open-source tools and techniques for building applications, adversaries and criminals are using the same tool stack. *How much we can quickly address our environment using DevOps principles? Then, we can go back and assess it quickly? Because today we can write a secure application, we are thinking, but tomorrow, who is going to go guarantee that the application is secure?*

Tim: *My library was secure today, but it's a known vulnerability tomorrow. How do I immediately evaluate and make those updates?*

Hasan: *When it pops up, how much quicker we can go and address those needs and update the patch or fix the box and as quickly as possible?*

Tim: The only way to do that is, really, through automation and embracing that.

Will: Sometimes in Agile parlance we say, *We want to develop and deliver a working high-quality product before the user has a chance to change their minds*. I think what you are talking about is, *We want to be able to deliver secure systems before adversaries have the insight about how to get in.*

Hasan: If you don't fix the system, it will be a zero-day vulnerability. It doesn't matter if it was released three months ago or four months ago. Today, at the moment, you are vulnerable if you don't change it.

Tim: Even if I know about the vulnerably, if I have a long accreditation package, I could fix the problem, but I can't get my authority to operate. So, I can't actually release my more secure release.

Will: With a lot of these programs, too, there are many systems that contribute to the overall performance of the program. Each of those subsystems has the potential to need an authority to operate. So, simulation labs and other places where we do field testing, there [are] ATO considerations there as well. The network that is provided by an ongoing attention to these issues is much more powerful than trying to fix individual blocks at the expense of time.

Tim: We talk about the rework and things of that nature, software assurance defects, security issues, and vulnerabilities. Studies have shown time and time again that if I can remove a defect early in the software development lifecycle, it's exponentially cheaper than waiting to the end. A vulnerability is just a very specific type of defect. The same principles apply. There is a return on investment. It is money saved if you just invest in the tools and get those defects out when I inject them not months or years later.

Will: I think it reinforces the notion, too, that good quality, high security comes about from careful attention to disciplined processes. It's not the addition of some magic sauce at some point in the timeline. It's doing things the way we're supposed to do them, as we know them, as professionals.

Hasan: I think there's a misunderstanding about Agile and Devops. People are thinking. *It's just chaos. It's just a cowboy programming, random thing.* Actually, Agile and DevOps bring principles and process. There is a discipline at work. It doesn't mean it's just [a] wild thing and you can do whatever you want, however you want it, change things randomly. It's not. The people are thinking it is constant changes, but there is a science behind it. There is a process behind it.

Mainly, for DevOps, there are principals about the integrated platform. There is a requirement focused on the developer, [who] has to commit messages linked to the case management system. There is an information order there, as well. So there is a discipline in the DevOps work, as well. Automation is also requiring discipline.

Tim: I actually think it is more discipline. My experience with DevOps and Agile fixes is it is more discipline, because if you don't do it right, it is so much more amplified that you're doing it wrong.

Hasan: If you don't do it right the one time, then you cannot automate it. If you make it right and commit automation, that means it is going to work all the time.

Will: There is no way to release working code every two weeks. And, unless you do it correctly, working code that meets our security standards even more. The tools really help us to be disciplined with less overhead.

Hasan: I completely agree, but if you have the tools and there is no process, the tools aren't going to work either. You have to have a mission understanding.

Tim: A tool for a fool is still a fool. It's true for any tool. To maximize the tooling, you have to have a plan. You have to have a process. You have to have consistent usage of those tools. It's really true with the DevOps concepts tools.

Hasan: A tool should be compatible with your environment and technology stack. Tools should be enablers for the people to make the process happen.

Will: One of the ways Eileen always talks about it is, *How do we have a low-drag approach to this*? I think that's a nice model to think about.

Eileen: So, I'm going to shift us a little bit. Tim, can you explain to our viewers a little bit about how the ATO process works?

Tim: Folks in RMF have a way of doing it. There are six steps that you refer to. First step is categorization. It's where you say, *What type of system do I have*? NIST has a whole process for categorization. It is based on the types of data that the system handles or processes, and the risk to the organization.

You figure out your categorization. Based on that, there is a whole list. There are hundreds of actual controls that have been published by the NIST process. It is anywhere from, *Thou shall have a policy to account management* to *You need to have multi-factor authentication* to *You have to have an emergency response practice.*

Hasan: NIST 800-53/53A.

Tim: There are hundreds of different controls that cover a lot of different things. Traditionally, what will happen is, based on the categories of your system, it maps to a specific set of controls. At a minimum, you inherit those controls. There are also additional things, like personal information, PII stuff, that give you additional controls above and beyond the core controls mapped to the categorizations.

Then what happens is, someone goes through all these controls that are applicable to my system. They have to document and explain how they are addressing that control in their system. They go step by step, one by one.

Once they explain how they did it, then an independent person has to come in and say, *Is this system actually doing what you said it is going to do?* Traditionally, they will do it very manually. They will say, *This control says this. Show me in the system that you're*

doing that. At the heart of it, it is very manually intensive. That is why they do a full ATO every three years, and they do an annual review.

Instead it should be, *This is what you say you are going to do. I ran some automated testing. Here are the automated test results that demonstrate that I have met this control. You can see how much quicker—as an auditor, I can just see a dashboard*. *I can just see the results of these tests, assuming that I agree that these tests do reinforce those controls*. There are still some manual controls that I will have to review annually, like policy-type of ones. In general, over half of them can be automated.

Will: Is there an analogy to test-driven development that you can draw there? There is a risk-driven development framework that can come about if we do it this way.

Hasan: That is an interesting development. You said a very key word: *documentation*. The biggest concern, especially for ATO folks that I have seen so far, based on my engagements, people miss documentation. Let [the] system create documentation. There is no way we can stop documentation. That is impossible. Why should the system create documentation for us? What the documentation requires, basically, is we are looking security controls. 800-53 depends on application. So, all this documentation can be auto-generated throughout the development pipeline.

Tim: That is what I meant by a dashboard. It very much is, *If I know what the controls are that I have to meet, I can write automated test cases, write scripts that validate that. I can map, This script goes with that control. When that script runs, every time I check in my changes—continuous integration, continuous testing constructs from DevOps—it generates test result*s. *When I fail, I have got to fix it now.*

Will: This can be demonstrated to an audience of people who have these considerations.

Hasan: If you take the security aspect right now, almost all industrial practitioners who think that DevOps works, it is adding automation for testing, like a functional testing, or like any ability-type of testing.

Tim: Those same tools that do functional testing can do security checking too.

Hasan: The issue, as I said, is there is no defined security requirements as part of function requirements. One is defined and says, *Here are my functional requirements. Here is the security portion related to function requirements. Here is applied security controls that will be part of my functions requirements*, which is basically the first

categorization of RMF and ATO process. Now the key is, *How can we document that process?*

If you have a fully integrated DevOps pipeline—all those components, like the wiki pages or the source-code repositories, case-management system, and continuous integration continuous deployment that comes after that. If you automatically test those things against the cases that you created, against the test cases that have been defined, then you are assured that it has been tested and then get the report directly. Still, humans will be in the loop, but the system will generate the documentation required that the person say *yes* or *no*.

Tim: Yes, then it is much less manually intensive. I just have to review the output, I don't have to actually perform all the manual validations, in terms of requirements. That is really the beauty of RMF versus the DIACAP process. RMF, if I knew [what] the categorization is and categorization is supposed to be pre-milestone B, so, it should actually [be] part of the contract. They should know what they are, at least initially, before contract award in the defense acquisition process. The developers have their requirements specification. The key is that the developers had to be given them as their requirements. They have to understand what they mean, and they have to work with security folks. That is the *Ops* part of *DevOps.*

Hasan: I'm going to play the developer's role and see if I am going to write an application. I have a look at my Scrum master sword and my Agile coach to say, *Hey you have to include this*. That's the agile that comes into the picture, whoever is running the show.

Will: One of the things we like to talk about in training classes is [that] many people work in an environment where we never have the time to do it correctly, but we always find the time to do it over. If we take the RMF on board at the outset, we are not doing it over. We're making the correct path the easy path, using the kind of tooling you are talking about.

Tim: From an Agile perspective, it is the minimally viable system functionality. Well, you can have all the greatest functionality in the world, but if I can't get an ATO it is useless. By definition, it is no longer viable. Part of that minimal viable system is security.

Will: I think there are a lot of practitioners in this space who are accustomed to being the bad-news providers at the end of the pipeline. To hear that viability of the system is

something that really hinges on this and to be able to operationalize that logic, I think, is a huge win for us.

Hasan: One more thing: one of the requirements that I believe the security folks know is, *What are the dependency libraries that are coming through my pipeline, not only for the application-related, functionality-wise, but also other requirements?* One of the good principles of DevOps comes in the big picture, which is infrastructure-as-a-code basis.

If you scripted all the dependencies, all the libraries as part of the application package, these things are not changing every time. But if you keep the versioning, like giving a word documentation and keep it as a code by itself, you can diff it. Instead of going for every library over and over for the re-approval process. Then you just look for, *What are the new libraries that I am going to use to just get approved for the pieces only, which is incremental approval for the dependencies?*

The big mistake I see so far is looking for a configuration management when they get their approval. A couple of years ago, I gave the huge list to one of our clients, which is more than 150 libraries. It took months to get their approval. Why don't we give them only an incremental version? They are not used to having an incremental version. Next time we gave them actual code, we literally gave them the code and said, *Here is our Chef script and compatible environment. We can diff it. We gave it to you six years ago. We only have a new version of libraries, and we got the approval process only for changes that we are writing based on the diff.*

It comes to trust, right? It is basically code running by itself, so you can compare it, and then only the changes would improve actual drop to one day. We dropped the code. They know that which version has been modified in terms of libraries. The next day they get back to us. *Yes it's approved. You can push it into the production environment.*

Tim: That's the power of infrastructure-as-code. It's because your script is your documentation. By definition, my documentation is up to date because the script won't work unless it's right.

Hasan: Which is verification isn't it? So I verified that it is going to work.

Will: I think there is another benefit to the mindset that comes from viewing infrastructure as code. It forces you to be much more specific and much more exact about what it is you are or are not doing. That mindset is really essential for all of these efforts that want to shift left the things that have typically occurred at the end of the lifecycle.

Often, it is not clear that people who are doing the work earlier in the lifecycle, what that activity is about. As we get much more conscious about it, shifting left becomes much easier.

Hasan: Shifting left is good because operational folks already using that have used that type of configuration management system. They already have the scripting environment. They are using it already. So, why don't we use that environment, which they're using in terms of the scripting? Pass it to the left, including with developer maybe going far left, and then the architectural pieces and going back to the acquisition pieces and ask [for] the requirements up front?

Will: What goes in [the] RFP, right?

Eileen: That actually brings me to my next question. If I am a brand new program manager, what do I do now? I'm new to our math. My contractor and my engineering organization are going to be using Agile and DevOps approaches. What should I do now?

Hasan: It's a tough question. I'll start it. Definitely, the program manager should understand their development environments first and what environment they are working on. It is really important to understand the technical needs and the program needs. You can start to establish a good platform, infrastructure-wise and platform-wise, which enables the people and developers and security folks and program managers and all the stakeholders [to be] able to see that platform. Then the developers will [be] able to use that platform actively so we can do automation on top of it.

Tim: I think it's a mind shift, a little bit, from a program manager's perspective. Traditionally, they take the DIACAP and the RMF stuff and say, *Who is my security specialist? Here, it's your problem.* The reality is it is an engineering problem. *How do engineers secure a system and build a secure system?* They should be taking the RMF and saying, *Engineers, it's your problem. Security person, you are responsible for validating that they are building me a secure system*

It is a very huge mind shift, but that's really what the program manager should be focused on. RMF is the system engineer's problem or the lead software engineer's problem. The security folks are there to make sure that they understand what the controls mean. They are the experts in security, but it is not the security person's responsibility to implement and build the secure system. They are the validator. They are answering questions. The developers need to own the problem. They need to solve the problem.

Will: In places where we are seeing Agile be successful, what we are seeing is people are being brought to the table. So I might have been part of an operational task that typically occurs at the end. I am coming up at the very beginning of the planning process and understanding, *What is going to be the impact to the infrastructure I need to operate when we have these kinds of groups of capabilities coming down the pipeline?* RMF becomes another system that constrains that process towards success. The earlier we can get those people involved, the earlier we can introduce those concepts, the better we are going to optimize the whole pipeline as opposed to optimizing some piece of it, or optimizing it with respect to some subset of attributes we care about. We want to have as many of those things simultaneously contributing to the decision-making we are doing at each point on along the way.

Hasan: You are saying, basically, *The problem is not my problem. It is not your problem. It is our problem*. That is a big mind-shift change: *it is our problem*.

Eileen: In five years, 10 years, what does *awesomeness* look like for the Risk Management Framework and Agile and DevOps?

Tim: I think all programs have a continuous authority to operate. There is no such thing as a three-year ATO event or a one-year, check-the-box event. I follow good acquisition Agile processes, and I build it in. It's a non-event. Security should be a non-event because it should always be there. I don't have a minimally viable product if it is not secure.

Will: So, a clean bill of health shouldn't be an exciting thing. It should just be normal.

Hasan: What I see for the future, things are changing a lot when I compare my background to 25 years ago. Where we are right now, and looking ahead for another 10 years, the programming concepts will keep changing. So, it is going to be more about configuration, more about automation.

Eventually, people will write only the configuration script. Eventually, they will develop a program using a Lego building-blocks-type-of approach. In that context, automation is a key principle, because it is going to be all configuration. It is going to be all about integration. If we can adapt our system quickly, then we will be winners, which is good automation. That's the future I see.

Maybe robots are going to write the code, eventually, or the system will write it by itself. It is going in that direction right now, especially for automation. There is a chatbot or

ChatOps in DevOps world. Basically, you can start up this continuous integration server with the one command line through the chat window. It is getting there. Then you can get the result back to your chat environment, move from the system, which is automated again. So, you can look at automation, to all this tooling perspective.

Will: I would add another dimension of ambition for us, if I may. I think the ecosystem that we are concerned with, where our products operate—we want to expand the scope of that ecosystem. In particular, I would like to see hardware, firmware, middleware—those kinds of things—be considered in this framework. Right now, there are not a lot of good solutions for, *How do we do rapid and frequent integrations that involve changes in hardware?* Software is perceived, not always accurately, to be somewhat easier because it is zeros and ones on the machine. But why can't we grow our focus here?

Tim: For the hardware pieces, from an assurance perspective, it is a huge issue. The days of the trusted foundries are no longer really in existence. *How do I assure that the chips and the capacitors, there is nothing hidden in there?* Also, a lot of chips now are actually programmed, or they are developed by programs. Most chip manufacturers, they don't actually manually design the chip. They use software. They put requirements. They diagram a design that auto generates how that chip should look and how that chip should be mapped. How do you assure that software that builds that hardware?

Eileen: Maybe you will all join me for a future conversation about exploring this in the hardware and embedded systems world.

Will: And we can bring all of our children, because they will be doing this work.

Eileen: Thank you all for joining us today to talk about this work. For our audience members, we will provide links to the resources mentioned today in the transcript for this podcast. This podcast is available on the SEI website at sei.cmu.edu/podcasts and on the Carnegie Mellon University iTunes U site and the SEI's YouTube channel. As always, if you have any questions or comments, please feel free to reach out to us at info@sei.cmu.edu. Thanks.