Walking Fast Into the Future: Evolvable Technical Reference Frameworks for Mixed-Criticality Systems

Featuring Nickolas Guertin and Douglas Schmidt as Interviewed by Suzanne Miller

---

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

Suzanne Miller: Welcome to the SEI Podcast Series. My name is Suzanne Miller, and I am a principal researcher in the SEI Software Solutions Division. Today I am very pleased to welcome to our podcast Nickolas Guertin, a senior systems software engineer with the SEI Software Solutions Division, and Doug Schmidt. Previously Doug served as the chief technical officer of the SEI, and he continues to collaborate with us on various projects. He is the associate provost for research, development, and technologies at Vanderbilt University and co-director of Vanderbilt's Data Science Institute, as well as the Cornelius Vanderbilt Professor of Engineering and professor of computer science. Today the two of them are joining us to talk about strategies for creating architectures for large-scale complex systems that comprise functions with a wide range of requirements. This is one of the most challenging areas in DoD acquisition, and so this approach and the strategies that they're going to talk about are very important to the future of our large systems. So, I want to welcome both of you today.

Nickolas Guertin: Thank you, Suz.

Douglas Schmidt: Thank you

Suzanne: And I want to, before we get into the topic, we always like to have people tell us a little bit about your backgrounds and the work you do here at the SEI. And things like what is the best part of your job. Doug, also please tell us about your role as a visiting scientist. That is an unusual role here, and so how does that play into your research life? So, I am going to start with Nick.

Nick: Thank you, Suz. It is a real pleasure to be with you here today. I have been here at the institute for four years, and it has been an amazing learning experience. I try to give as good as I get, but I'm pretty sure I'm short on that exchange. As a systems engineer, I work a lot with a variety of different programs where they are thinking about acquisition strategies but then also helping them with understanding how to do their technical approaches. Certainly I have gotten a lot more comfortable and familiar with things like DevOps and Agile, while we are also trying to help the Defense Department see their way through these new acquisition authorities that are coming out of things like the Software Acquisition Pathway. I am a huge fan of that. One of the big jobs I got to do was helping Forrest Shull and the work he was doing with the Defense Innovation Board, and that was a fantastic experience. That report ended up getting turned into real policy. It was just an amazing thing to do.

I had the pleasure to work with Doug again. Doug and I worked together when I was a Navy acquisition guy working in the Pentagon. While I was there, I was also doing some work under the Better Buying Power initiative with the other services—helping understand how to do things like modular open-system approaches, artful acquisition strategies, new competitive dynamics, bringing in a variety of different businesses to create modular capabilities that can be composed and delivered more effectively and fluidly. Some of those policy initiatives are starting to come to fruition too. It has been great to be a part of moving this big machine. It takes patience, but sometimes that patience is rewarded. I have also worked on things like a variety of different submarine systems, weapons sensors, combat management, and also as a Navy Reserve engineering officer, did what we call pump kicking, working on ship structures and hull, mechanical, and electrical systems. Even before that, I was an enlisted power-plant operator on a couple of different submarines at the peak of the Cold War up in the North Atlantic, doing a bunch of fun stuff that I can't talk about here.

Suzanne: All right. So, Nick in addition to your background that you've given us, we know that you were recently nominated by President Biden to serve as director of Operational Test and Evaluation for the U.S. Department of Defense. If confirmed in that role, what are some of the experiences that you've talked about and other experiences you've had that make this an exciting shift and transition for you into a Department of Defense-level position?

Nick: Well, Suz, thank you very much for a great question. I have got to tell you, this came a little bit as a bolt from the blue. I didn't, was not looking for a new job but when called to serve,

I was super excited about being able to serve in this capacity and take on the kind of challenges that will be faced in the future. Some of my experiences leading into this on things [included] working on a variety of different systems, sensors, combat management, weapons. I have done a lot of work in development tests and some work in operational tests, but then also having some experience with the SEI especially, thinking about how do we build systems in the future so they're changeable over time. And then we've got to switch that over to say how are we going to test these systems over time? What does the operational test community do when something is constantly in motion? I don't have the answers for that yet but we have some of the background to help inform what activities I might do with that team as we think about that and work on it together.

Suzanne: I just want to say for our viewers that have not had a chance to meet Nick in person, I did get a chance. I had a chance to work with him down in his D.C. office a couple of times, and he probably has the coolest collection of coins—the coins that people give out in the DoD for accomplishments and achievements—of anybody that I have seen. I know that what he has done has been important to a lot of different groups, not just to the SEI. Doug, I know you don't have the same collection of coins that Nick does, but give us a little bit of your background and especially tell us a little bit about what it is like being a visiting scientist here at the SEI.

Doug: Sure, you bet. Thank you very much for having me on. I am a professor of computer science at Vanderbilt University, and I wear other hats here as well. I really enjoy teaching courses on mobile cloud computing. I have been doing research on cyber-physical systems, machine learning, machine learning applied to healthcare, topics related to the blockchain, and many other research topics for many years.

But I also have another passion, and that is to work on real-world, large-scale systems and systems of systems. And so since 2005, I have been involved with the Software Engineering Institute in many different capacities. I was part of the original ultra-large-scale systems study that was done in 2005, which had a big impact on the research community. As you mentioned earlier, I was the chief technology officer and deputy director of research at the SEI about a decade ago, and I continue to work with the SEI in various ways, primarily working on various, what are called PWP, program work-plan projects, where we work with real projects. We work with different services. We help to build the nation's defense capabilities and other capabilities to make sure that the software-reliant systems we depend on are going to be affordable and effective and resilient. That is really a passion I have. One of the great things about working at the SEI is I get a chance to work with people like Nick and other folks who have a very long history of serving our country and really pushing the boundaries of what is possible with software technologies. That is one of the things that gets me so excited about continuing my

relationship with the SEI as a visiting scientist, even though I am no longer in the chief-technology-officer role.

Suzanne: Thank you, Doug. And I know that many of us get to interact with you and benefit from that experience that you have in both the research area and in the DoD acquisition area. Good to have you both on board today. So, let's talk about the work on technical reference frameworks for complex mission-critical systems. That is the topic for today. Doug, why don't we start off by having you explain for us what a technical reference framework is, and then discuss why complex systems comprising functions of mixed criticality—we'll talk about that a little bit more—what are those systems and why do they need TRFs. So what problem are we trying to solve with this?

Doug: As I mentioned before, many of the systems we work on at the SEI are very large scale. They are very complicated. They involve decades of service. They have to be refreshed and updated. Think of things, for example, like the B-52 bomber, which has been in service since the 1950s, and the old cliche is that there will be pilots flying the B-52 in service who haven't even been born yet. So these are systems that live for a very long time. Obviously it is extremely essential to get the architecture right. If you don't get the architecture right, then you will spend all your time in the sustainment activities trying to fix bugs and retrofit capabilities, and it becomes very expensive. You end up with less than you would like.

A big role of these systems of course is to get the architecture right. One of the things we are discovering as we are working on larger and larger systems and systems of systems, is that these systems of systems are actually composed out of many other pieces, systems, or subsystems. And those different systems and subsystems have different needs, different requirements, different technology bases. And so one of the things we are doing with the concept of a technical reference framework or technical reference frameworks, plural, is to try to make sure that we have the right framework being applied for the right part of the system.

Just to give a very simple example: Imagine you are building a combat system or an electronic system for an aircraft carrier. There will be parts of that system that require closed-loop control, things like the close-in defense system. For those things, you are going to need very tightly coupled cyber-physical systems that are very responsive, systems where the right answer delivered too late becomes the wrong answer. So you need real-time capabilities, very high levels of security and safety, and so on. There will be other parts of that type of a system that are more command-and-control oriented. So people who are tracking incoming threats, engaging with different targets, manning different parts of the ship, of the aircraft carrier, and so on. And there will be a different set of requirements there. Some of the requirements for closed-in, closed-loop control may be a little bit less stringent, but you also have to deal with a lot more things to keep track of. And then there will be other parts of the system which might be things

like the crew-entertainment system, where you are going to be basically doing enterprise-style computing.

One of the things that Nick and I will talk about today in our podcast is the need to have the right architecture or architectures for the different pieces of those systems. So the kinds of languages, the kinds of operating systems, the kinds of hardware, network protocols, the kinds of middleware, and so on that you would use for the cyber-physical pieces of, say, the closed-in weapons-defense system, is not the same that you would necessarily need for the entertainment system. And therefore you need to make sure you pick the right kind of technologies, the right kind of architectures, the right pieces of the system. If you do this well, the systems can grow and evolve more effectively, and also they can be certified and verified for various compliance regimes because not every piece of the system requires the same degree of scrutiny and the same degree of compliance conformance.

Suzanne: Thank you. And Nick, did you have anything you wanted to add to that in terms of mixed criticality or the role of technical reference frameworks?

Nick: Thanks, Suz. One of the things that I worked on when I was in PEO Submarines was a different lifecycle-management approach for like a sensor suite. We had this battle rhythm where we were changing out the hardware on…We were looking at different hardware every two years, and no one ship would get anything older than say like six years old as a different lifecycle management approach. But the underpinnings for that required some of the things Doug talked about, like middlewares and modular construction, and also the acquisition lifecycle was different. We had different buying plans for different pieces of hardware, different installation teams, and had to roll around to different ships. But then when we looked at like the weapon system where those certification boundaries were a lot tighter, the temporal range of time for what it took to deliver a new capability was a lot longer, and the need to do hardware refreshes was just plain different. To apply the same practices for doing the sensor system, which could be a little bit looser in terms of time, was dealing with a lot more large amounts of data and human-scale analytics were very different than safely launching energetic systems. So, we wanted to make sure as we were thinking about this. How do we look at that and evolve it in the right pattern instead of just doing a one-size-fits-all approach regardless of what the actual problem set was on the other side?

Suzanne: I want to reference something you said there because it has been in my mind in some of the systems that we work on together. That evolvability. We think about software being very evolvable and malleable. That is one of its reasons for having so much play in these systems. But you are talking about technical reference frameworks not just for the software but for the architecture of the entire system, which means you are dealing with hardware issues as well. And that technology refresh, the pace, when we have to talk about a B-52 or something similar, the

technology, just the materials that are available today are different than when the B-52 was first envisioned. So that evolvability aspect, I think the technical-reference-framework aspect. Am I correct in thinking that that's really one of the critical places that you think about, *What are some of the scenarios for evolvability that are going to be needed for the various elements?* I'll take your example, Doug, of the crew-entertainment system. If I have to replace that whole thing out, so what. But sensors? Those are a little more expensive and some of the other really specialty things, you just can't say, *Oh, I'm going to buy a new one because we have something new available*. So, talk for a minute about the importance of evolvability and how the technical reference framework helps with that. This time I will start with Nick.

Nick: So, you were talking about the crew-entertainment system. Well, what if we don't just chuck the whole thing and bring in a new one? What if we just bring in the new parts we need? That idea would drive a weapon-system person nuts. They have to have tight configuration control, certified deployment. The idea of mixing and matching stuff because it works is not the right business plan. It's not the right systems-engineering approach. So you need to match up the technical domain and the problem set, but you don't need an infinite variety. You need just a few.

Like Doug talked about earlier, you need like the real-time missions-critical stuff and then way on the other end maybe is kind of like the large-scale data-analytics infotainment or other kind of big-data stuff. And maybe one in the middle like a command-and-control thing. So as we were thinking about this, we are like, *Well, is it an infinite scale or are there maybe just a few with appropriate levels of overlap?* Because otherwise it would just be too complicated, right? You have to find a way to simplify the problem a little bit.

Suzanne: Do you want to take that away? I bet you have something to say about that, Doug.

Doug: Sure. I think when you talk about evolvability, it's very important to keep perspective here. One of the things that gets me so excited about working with the SEI is the ability to work on systems that are going to matter for a very long time. Some of the systems that Nick and I and our colleagues are working on are expected to last 75-to-80 years in the future. And if we roll the clock back 75 or 80 years, we are basically at the end of World War II. And imagine if we had to live with the constraints for the systems we are building today based on the technologies, based on the processes, based on the languages—there weren't really a lot of computers in those days—that were from the late 1940s. Can you imagine how impossible it would be to be successful doing today's missions with today's threats and today's capabilities if we were hamstrung by decisions made 75 to 80 years ago?

So that is really what we are looking at here. We are trying to make sure that as we move forward in this brave new world of sequestration and defense funding cuts and so on that we

sometimes go back and forth on and certainly new threats, that we are not going to put ourselves in the Procrustean bed of solutions that don't evolve over time. And looking back to what Nick was just saying about the different domains or the different technical reference frameworks, the kinds of decisions you need to make for tight-loop, closed-loop, real-time embedded systems, are very stringent, and yet you still have to be prepared for the inevitable evolution that will take place in that domain over the next 80 years. And if you take a look at things again about soft real-time systems, command and control, the data analytics, the human-time systems, the enterprise-scale support tech systems, those things are going to be changing very rapidly. So we need to make sure we pick the right architectures that are capable of expanding and contracting over time to meet the needs at that time without limiting ourselves to things that we can think about today but [that] will inevitably change over the next few decades.

Suzanne: So there are two sides to this that I want to bring in and relate the evolvability to. One of them is interconnectivity and functional interoperability. You mentioned systems of systems, and we are really in that place. And so what are the challenges there? But I also don't want to forget what Nick talked about earlier: How do all these ideas affect our acquisition lifecycle? Because when you talk about the decisions that were made 75 years ago, in our current acquisition process, that would be considered a technical baseline. And from an acquisition viewpoint, those of us that have worked in that arena know that once you establish a technical baseline, it is not as easy as we might need it to be to change that. So we have some things in place in the way we acquire things, and we have some technical issues in terms of connectivity and interoperability. What are some of the things that the technical reference frameworks help us deal with those two kinds of things with? And Doug, why don't you go ahead and start with the interconnectivity, and then Nick can follow up with the acquisition.

Doug: Sure. Whenever we think about long-lived systems, we have to think about issues like portability. We have to think about issues like interoperability. And we have to think about issues like integration and basically testing and validation and verification. From a portability point of view, nowadays we have a much better position to start. If you want to build systems that are going to last for a long time, we probably want to build them as much as possible on an open infrastructure. We have things like POSIX. We have various types of middleware. We have virtual machines that have been well established with all kinds of test compatibility kits to make sure that they are portable, and you can relatively do write-once-run-anywhere kinds of capabilities. So those baselines are important from a portability point of view, and that is really important to get right.

The other thing that is important if you want to make the system capable of being interconnected, as you were talking about, Suz, is picking good protocols and then making sure that those protocols are protocols that can evolve over time so you don't get locked into yesterday's

standards, which is a big problem that we often run into. Making sure that the protocols and the APIs and the implementations of those APIs and portability layers are things that are living and capable of evolving and extending. Just as we don't want to get locked into proprietary vendor solutions, we don't want to get locked into obsolete standards. So, there are various ways of thinking about layering and abstracting. There is also a whole focus these days on model-driven approaches where you synthesize lots of the system rather than having to write it from scratch. So those are all key themes that people need to be doing for the practice of good software and good software engineering over time if you want to build for the future.

And I think another key thing that I am sure we will talk a bit more about is making sure that we set up testing regimes where you can be confident that what you are releasing, either on a baseline-like basis or block-style basis as the DoD often does, or in a more continuous style, which is what the commercial world is doing increasingly, we want to make sure that we're using the power of computing to test things, to have good test-driven development, good continuous integration/continuous deployment, good DevOps principles, so that we can be very confident that what we are putting out there is going to meet the requirements that we have put forth. Historically the DoD has struggled with a lot of that, but one of the things we are learning with all of our work in the commercial world and our academic research at the SEI is the fact that it is much easier now if you pick the right sets of technologies to have an infrastructure that is capable of producing software that can be known to work and have higher confidence it actually delivers on its requirements, both functional and non-functional. Those are some of the themes from an evolvability point of view. Pick the right APIs to get portability, pick the right protocols to get interoperability, and then set up a testing environment where you can continuously evolve and make sure the system works as the way you expect it to work relative to the requirements.

Suzanne: And what I am intuiting from your answer is that these are elements that would be included in the technical reference framework.

Doug: Yes.

Suzanne: In terms of, these are considerations that we want to make sure are known and made explicit. Because that is really the purpose of a framework like that is to make things that are in our heads explicit to the people that follow us. Because we are going to retire at some point. Even Nick is going to retire at some point.

Doug: Well I think the other key point there, to follow up on that, is that for different technical reference frameworks, you may choose different APIs. You may choose different protocols. There is no shame in that. One of the mistakes people often make is they try to come up with a one-size-fits-all architecture, and then they try to have it span everything from the closed-loop

control systems all the way to the crew-entertainment system. And inevitably somebody pays the price. Either you make something that is very, very real time and very predictable but doesn't scale, or you do something that is going to be very scalable but won't meet the hard real-time requirements. So, picking and choosing intelligently is really what this technical-reference-frameworks philosophy is in a mixed-criticality and systems-of-systems world.

Suzanne: Excellent. All right, and Nick, so just thinking about what Doug has said in terms of everything from testing, protocols, all of those sorts of things, talk about the acquisition lifecycle a little more, and what are some of the implications of using technical reference frameworks, and how do we need to think about evolving our acquisition system to be able to take best advantage of these concepts?

Nick: One of the things that came out for me in ways that I had lived in different acquisitions I was involved in, but was really cemented in my mind that came from the Defense Innovation Board was, you know, software is never done. Even if you build the perfect system and you nailed every single requirement right, and your operators or users will never want a single new thing…Wait for *that* to happen!…you are going to end up dealing with cyber vulnerabilities, and operating-system updates, and deprecating tools, and all the rest of that stuff. It is constantly in motion.

Software-development methods are constantly evolving. I don't think we are really going to be in sustainment in the way we have characterized it in the past. It is more about, sustainment can't be about standing still with what you have; it has got to be about walking fast into the future. Everything we build has got to be thought about in the context of walking fast into the future.

And then interoperability is kind of a part of that too because as these systems evolve, they might need to exchange some different information or consume some new information. And so as user needs evolve, as they express new capability that would need to be added to things we have, we need to think about how that impacts the interoperability aspect as well. So I have this kind of nested *if* statement, like, if a lot of the things we build are software reliant and that a low-friction way of delivering new capability or upgrading our cyber posture is through software updates. And that is all interoperable at a different level than we experience in our home lives, right? Our aircraft need to talk to our ground systems, need to talk to our sea systems, and all of those things are much more interactive, dynamically interactive, than what we see in our normal day-to-day lives. So we need to start thinking about interoperability as a continuum of change. That is just another aspect of how we need to re-characterize the way we acquire these complex and evolving mission systems.

Suzanne: OK. And I will say that I, too, am a fan of the Adaptive Acquisition Pathway. One of the things that I have been thinking about is where can we take some of the concepts that we've

used in the software-acquisition pathways and bring them up to the systems level. That is the iterative aspect. Doug mentioned that model-based engineering is one of the things that when you start getting really good physical models, which we can do today, you can actually start that synthesis of hardware-software integration through models. Is that something that is accounted for in the technical reference frameworks? Is the role of modeling in that sort of evolution, or is that something that is for future research?

Nick: I would like to jump in on that. I think that there is a lot of work to be done in that area. One of the things that I have become aware of more recently is that our mission systems are growing in complexity, and the kinds of threat environments they have to respond to aren't the sort of things we can go to just one place and just test everything. So the notion of having a model of the system as well as of the threat environment and combining that with actual testing, physical testing, might need to be a part of our future for a lot of things, if not everything. And so, I think there is a lot of research to be done in how do we include models that have been tested and used, perhaps a term, *accredited*, that trust that models accurately reflect what it is they are supposed to be presenting. Along with system testing and maybe even how do we lash different test environments together, I think there is a lot of work to be done there, and it should be a pretty exciting time.

Suzanne: Doug, did you want to comment on that?

Doug: Well, Nick's really the acquisition expert here. I guess what I will say is that one of the challenges we've been facing in these areas for many years is that people who do large-scale major defense acquisition systems have historically thought of software as an afterthought. And one of the things that is becoming clear is more and more capability is being done in software. And so we really have to raise the bar. We have to educate people at all levels. We have to educate people who are program managers and PEO executives and military officers who run these systems. We have to work on the certification community to make sure everybody up and down the chain understands the strategic importance of software, number one, and perhaps even more importantly, knows where to turn to get help because not everybody is an expert on those topics. There are lots of people in various parts of the commercial industry who are very good at some of these things that don't necessarily understand the DoD problem space.

There are a lot of people in the DoD space who understand the hardware platforms very well, but they wrestle with software, and they certainly have to think more carefully as to how to adapt and adopt new capabilities that are coming online, things like software factories for example and product line architectures. The commercial world is doing a pretty good job of that in many cases, but the DoD still has a long way to go. Part of the challenge has been that up to this point, there really hasn't been an emphasis on the DoD having software as a key core competency. That has got to change. That is a crucial ingredient for future success if you want to be successful with

the software-reliant systems that we are trying to put together to last over the next couple decades.

Suzanne: I know Nick mentioned the DIB study, the workforce development, digital workforce development was one of the key themes in that study, which I have seen in different ways. That is a topic for another podcast. I definitely agree with you that bringing in knowledge about software throughout the enterprise is one of the ways that we are looking at increasing that software competency to be able to deal with things like technical reference frameworks and complex interoperability, differences in protocols, and things like that.

I want to switch at this point to talk a little bit about transition. In the podcast we like to talk to people about what is available to program offices, contractors, people that would be interested in this topic, to help them conceptualize, acquire, use the frameworks. What kinds of things do we have available to them to help with this problem of evolving systems that are this complex? I will start with Nick this time.

Nick: Well, thanks. I have been involved in some standards work especially with the Open Group for a consortium on future airborne capability environment. It is a reference framework especially tuned toward real-time cyber-physical aspects of what it takes to fly an airplane.

They have spun out another reference framework for sensors. You know, how do we put different eyeballs and different radars on aircraft. More recently, there has been just beginning work on electronic warfare. What is a reference framework for electronic warfare? Those are three very different technical domains in terms of what they do, and one size does not fit all. I think some of the other work that is being done like with the Air Force with Platform One, and the Navy and Army is trying to replicate those kinds of things for more the sort of things you can do with virtualization, hosting, as well as containerization, things that are more geared toward command-and-control and data-analytics kind of stuff. Those are fantastic places to get everybody to participate in creating the environment that is going to be well suited for a lot of different things, but not everything. And to start to maybe collect a little bit more on what it takes to build out these things instead of having every system as its own special snowflake.

Suzanne: Gotcha. Doug, comments from you on transition resources for people that want to know more about this?

Doug: Sure, you bet. So the good news is especially for the technical reference framework portions that are more on the enterprise-system scale, again the crew-entertainment system or the data-exploitation-style systems, the big-data systems, there is a lot of great technology that is available out there off the shelf, oftentimes free of charge, open source that are widely used in that space. I think the enterprise space is pretty well covered. You don't have to really be a

specialty group to get access to that capability. There are tons of consultants, tons of resources, books, videos, YouTube, and so on to help you.

As you start moving closer towards the middle TRFs [technical reference frameworks], the command-and-control portions, and certainly when you get into the closed-loop control systems when you are getting into real-time embedded systems, a lot of the stuff that is available off the shelf is not necessarily going to work in those spaces. The good news is there are a lot of advances that have taken place there over the past few decades, so we have things, for example, like obviously POSIX for the operating system layers, things like the OMG [Object Management Group] Data-Distribution Service that is very well established, very well defined, and has proven itself time and time again to work in that space.

I think one of the things that is really fascinating to me and why the SEI is such an interesting place to work is the fact that just about the time we figured out how to do transition of this classic modular open-system-style technologies, we are getting that to the point where it's technology readiness level [TRL] 7 and 8, all of a sudden the world shifts and now we're moving to AI-based approaches and machine learning where we are TRL, technology readiness level 2 and 3, much more in the early stages. And so I think one of the real challenges we have is, as the whole DoD shifts towards intelligent systems of various flavors, we are really starting over from scratch in a lot of ways about how to develop those systems, how to assure those systems, how to make sure that we know how to use those techniques to build systems better.

There is lots of work that is going on at the SEI and elsewhere on AI-enabled software engineering. And then likewise, as people start to use AI algorithms and machine learning and transformers and deep-learning models to actually run the systems, there is a whole new set of opportunities for thinking about how do we transition work on research for assuring those kinds of probabilistic systems. That is really, I think, where the exciting next generation is. How do we do the research on these topics, how do we get breakthrough in results, and how do we transition them. So I think we are doing a fairly good job for today's systems but tomorrow's systems are going to be much more dynamic, much more adaptive, much more non-deterministic. A lot of the old techniques that we have been relying on for decades for our safety and our security and so on are going to have to be rethought very carefully with leveraging advanced research.

Suzanne: Sounds like we have some future podcast opportunities. All right, any closing remarks from either of you before we wrap this up?

Nick: No, Suz. Thanks. You have done a great job today. I really appreciate the opportunity to get this work that like Doug said earlier, something that we have been noodling on for quite some time now, and some of it is maturing to a point where it should become just the standard practice. Everybody should be appreciating that. Don't go it alone. There are plenty of tools out

there, lots of different ways of building out the environments that you need without having to create it from scratch. And there are people out there that can help you figure that out.

Doug: I will throw a couple other things in real quick. In my mind one of the keys to success here is to have a holistic view of developing these next-generation systems. We need to think about architectures of systems of systems. We need to think about technical reference frameworks. And one of the big, long poles in the tent is how do we make sure that we are in a position to certify and verify and validate these kinds of systems. Because that is often the limiting factor in practice. There's lots of great research that's been done on things like dynamic resource management and adaptive this and adaptive that and machine-learning this and machine-learning that, but until we get to the point where we have ways of being able to increase confidence that those systems are going to work as expected when it counts every time, we are going to be very hard-pressed to get them rolled out in practice.

And one of the things that makes this tricky is a lot of our adversaries don't hold themselves to the same safety and security standards that we do. So it is going to enable them to move a lot faster than we can if we're not careful. So we have to leverage the new technologies that are coming out, new advances in cybersecurity, new advances in machine learning. We have to have a way to harness those things so we can stay above the competition. I think again, things like model-driven approaches are crucially important there because we don't have a workforce that can do all these things by hand anymore. The United States just doesn't have enough organic U.S. citizens to build the defense systems of the future. We are going to need to rely on automation. Advances in automation and testing and verification and safety and so on are absolutely essential to keep ourselves one step ahead of the adversary.

Suzanne: Doug, Nick, I want to thank you both for such an insightful discussion. I have really enjoyed this. Before you leave though, I know that some of our viewers are probably wondering about your backgrounds. I know what the background is for Doug as I well should, but why don't you tell us what that background is for people that don't know as well as I do what it is.

Doug: This is the software engineering building at Carnegie Mellon University. If you take a look and stand in front of the building and you look at the reflective glass, you will see the Saint Paul's Cathedral that's right behind it. For a number of years, my office was on the fifth floor of the SEI building and so I had a chance to look out every day at this beautiful cathedral, which is a wonderful example, by the way, of architecture and the beauty of following patterns and ways of making things work and be stable to live over long periods of time. I have always enjoyed that view and so that's what my background is representing.

Suzanne: Now Nick, your background I would have to guess what your background is. So why don't you tell us what your background is.

Nick: It is the north pole of Saturn. And nobody knows why it has a hexagon shape on the front of it that is like as wide as like three Earths. But it is cool looking.

Suzanne: Very cool looking.

Yes. well, thank you. All right, so thank you both again and for our audience, I want to make sure you know that we will include links in the transcript to resources and some ideas that were mentioned during this podcast. I know not everybody knows what TRLs are. We will put some links in for that. I want to thank you again for joining us, and I hope you get as much out of this as I did.

Doug: Thank you.

Nick: Thanks.

Thanks for joining us. This episode is available where you download podcasts, including SoundCloud, Stitcher, TuneIn Radio, Google Podcasts, and Apple Podcasts. It is also available on the SEI website at sei.cmu.edu/podcasts and the SEI's YouTube channel. This copyrighted work is made available through the Software Engineering Institute, a federally funded research and development center sponsored by the U.S. Department of Defense. For more information about the SEI and this work, please visit www.sei.cmu.edu. As always, if you have any questions, please don't hesitate to email us at info@sei.cmu.edu. Thank you.