Software and Systems Collaboration in the Era of Smart Systems

Featuring Paul Nielsen as Interviewed by Suzanne Miller

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 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.

Suzanne Miller: Welcome to the SEI Podcast Series. My name is Suzanne Miller, and I am a principal researcher in the SEI's Software Solutions Division. Today, I am joined by the director of the SEI, Dr. Paul Nielsen. We are here to talk about the growing need for the disciplines of systems engineering and software engineering to develop effective ways of collaborating, especially in the era of what we call *smart systems*. Welcome, Paul.

Paul Nielsen: Hi, Suz. How are you doing today?

Suzanne: Good, good. Glad to be here talking with you. Let's start out by you telling us a little bit about yourself, because not everybody knows you. And why as the director of a software federally funded research and development center [FFRDC], you feel it's important to be discussing the systems and software engineering collaboration in relation to smart systems.

Paul: Sure. OK Suz. First of all, I've been here at the Software Engineering Institute for about 17 years now. We are an FFRDC that concentrates on software engineering, cybersecurity, and AI [artificial intelligence] engineering. But before I joined here, I was in the Air Force for 32 years, mostly working on space systems, command-and-control systems, and science and technology.

In my job in the Air Force, I was really more of a system engineer most of the time, although I am trained as a physicist. Now, of course, here I am, in the Software Engineering Institute. It just struck me from the very beginning of arriving at SEI, how these two disciplines of software engineering and system engineering had grown apart and yet had so much in common and could really benefit from learning from each other. One of my own personal initiatives while at the SEI has been to try to bring those two disciplines closer together.

Suzanne: This is not a new issue. When I joined the SEI in 1993, one of our first activities was actually helping to formulate some guidance on systems engineering in relation to software. This is something that, as you say, it sort of waxes and wanes in terms of interest and importance. But we are now moving into kind of a new era I think that we want to talk about, which is the smart systems. Let's talk about that trend towards smart systems and how that leads to this convergence of systems and software engineering.

Paul: Sure. Well, as we move into this era where we have more aspects of artificial intelligence in our systems, it becomes even more important that we architect and engineer these systems well. In most cases, these systems end up being more complex and more interconnected than systems were in the past. The whole movement to these larger-type systems and interconnectedness really is one of the rationales for why system engineering and software engineering got started. Because at some point, you have to break the system down into modules, or components, or subsystems as system engineers would say, and then build them back up. And so both of these disciplines have done this in the past in various ways. But now, we are seeing that the systems we are building now are never all hardware or all software. They are software and hardware systems. Sometimes, as you know, we call those cyber-physical systems. In addition, sometimes now, many of our systems also have aspects of social merit in them, so they become socio-technical systems as well, which even adds another level of complexity.

Suzanne: These are words with lots of syllables, but that is because…. Cyber-physical, socio-technical systems, that describes a wide range of aspects. I would argue that that is actually one of the key reasons that systems engineering and software engineering need to collaborate because that cyber part is software, that physical part, systems engineering has traditionally dominated. Then you get into the socio-technical, and both disciplines have to deal with that social aspect. So working together makes perfect sense in this kind of world. What are some of the opportunities?

Paul: I was going to say, one thing Suz is that if you go in the past, as always, life was simpler in many of our youths, and engineers worried about engineering things. But now, we find that that is not enough. Whether we are worried about equity or climate change, or regulation, or fairness, all of these extra things come in. We are finding that as engineers, to do our job right, we have to consider them as well.

Suzanne: What are some of the opportunities that may be different from the past of achieving synergy between systems and software engineering that you see?

Paul: Well, I think there are lots of them, obviously. One of the key things for any discipline is that it usually brings some ways of looking at problems, examining problems, and tools. And different disciplines always have some different tools and some different views. When you get to systems that are highly complex, man, as an engineer, you need all those tools you can bring to the table. And so by bringing these two systems together, we bring more tools to the table that allows people to solve hard problems. In addition, just as we always talk about diversity and inclusivity, when we bring people that are trained as software engineers together with people that are trained as system engineers, we get the best of those worlds together, too. As you know, many of our systems today, a lot of the functionality rests in the software. We have physical parts of the system too, but a lot of the functionality rests in software. So we need those different views to make these systems work.

Suzanne: I would argue that systems engineers and software engineers, in my experience, visualize the systems differently, and that difference is one of the things that we need to leverage. Because when I have software engineers that are all about the data, and where the data is coming from, and where it's going, and the hardware is often the transport vehicle for that data. That is a different view of the world than a hardware view of the world which is, *This component fits inside of that component fits inside of that subsystem*. So being able to marry and get advantage from both of those views, which are a little bit orthogonal, can really influence things like your architecture, right? We want evolvable architectures. Hardware is harder to evolve in some ways than software, although we can make software difficult to evolve too.

Paul: Yes, we can.

Suzanne: But, having that viewpoint of, *I want something evolvable* is one of the ways that we start taking advantage of each other's disciplines in what I see.

Paul: Yes. I think evolvability is maybe one of the keys, we would call it quality attribute in the software engineering world. Maybe the system engineers would call it a *non-functional requirement*. We are finding that one of the best ways to make really good systems is to keep evolving them, to keep improving them, not to have a static endpoint and push them out of the door, never to be changed again. That is also a real hallmark of something that was developed in the commercial software world of DevOps and continuous development, integration, and delivery, and deployment. That is a little more stressful for the physical side of things, for the hardware side.

But in software, the commercial companies have brought this almost to the limit of upgrades many, many times a day. Like I said, a little tougher on the hardware side. But I think the hardware side is trying to learn from the software side in doing this continuous evolution in a different way than they have done in the past.

Suzanne: So that relates to one of the government initiatives that has taken root in the last few years, which is digital engineering. The idea that, as part of solving that problem of making the hardware more evolvable, we do more of our analysis and real engineering of *What are our options* and *Which ones should we close out now* and *Which ones should we experiment with more in modeling environments*. We are actually starting to talk about ModDevOps. We have got modeling, dev, and ops together. Is that one of the places that you are seeing some of this opportunity for kind of bringing the engineers together? Because the systems engineers tend to be very comfortable with these kinds of modeling techniques. They can help us to understand some of the perspectives that way.

Paul: Sure. Both disciplines had really jumped into the model-based design-analysis kind of field. What we are finding is that the ways both disciplines were doing this were very compatible. Through model-based design, you can iterate design much earlier, you can look at alternatives. You can examine not only the quality of the system you are going to build, but in some cases, you can get quantifiable results too for all kinds of things like performance and throughput and latency and such. These can be very important in a system sense. In the software engineering world, the DevOps initiative is one thing, but then, the model-based software engineering is another. Right now, we are looking at ways to bring both of those to bear on the very complex problems that we are seeing.

Suzanne: We have done podcasts on model-based engineering before, I want to remind our audience. When we talk about model-based engineering, we are not just talking about model-based specification. We are not just talking about representing just the requirements in a model. We are talking about using that model to analyze effects of making changes, effects of switching out components, effects of changing architectures. That is when we really get the power of the model-based engineering into play, which I think our systems engineering brethren are very excited about.

Paul: Well in some ways, these models can have enough fidelity that they can actually lead to self-generated code, ultimately. They are almost like a higher-level language in a way; these models can be that good that they are a higher-level language.

Suzanne: What are some other developments that indicate progress to the convergence of these disciplines that we have been talking about? We have talked about model-based engineering. What are some other indicators for you that we're getting farther along with this?

Paul: Well, one of the most basic ones is that both communities seemed like they wanted to talk, which is a good start, right? Just like, you can have people who speak two different languages, we early on found we had different languages for certain things. Sometimes we were talking about the same thing but using different words. We had to come together and start to understand each other's vocabulary and such. One thing that has been very encouraging to me is that the professional society for system engineers in the country, INCOSE, International Council on Systems Engineering, has been reaching out to us for some time to try to bring this marriage of software engineering and system engineering together.

About four years ago, they started an initiative where they established a working group to look at these interfaces. Of course, we had several people working on that, especially Sarah Sheard that worked on that quite a bit. Then, even more recently than that, this society, INCOSE, is doing a Vision 35, what is system engineering going to be like in 2035? I have been working on that document with them. It has lots of aspects of how system engineering and software engineering have to work together. I think it's a growing awareness of the strong amount of content and functionality that resides in software. It's not enough just to balance the hardware part of the system now, you have to have everything balanced. System engineers for years have done this allocation of requirements, allocation of functionality. At times, that was to hardware subsystems but now, they are realizing that the software itself is bigger than any subsystem, so they have to look upon it as a more major constituent of the system itself.

Suzanne: And it's cross-cutting.

Paul: Yes.

Suzanne: One of the things we talk about in some of our classes is that the mental model of hardware engineering is, *This is a part of that*. The mental model of software is, *This, and this, and this, and this, use that*. So it's not the same. It's not a structural view. It's more of a usage view. That alone creates a different way of looking at the two systems. I know in my case, when I have been able to get the systems engineers and the software engineers to talk about that, we get a much richer conversation about, how do we deal with some of the software architecture issues that you are only going to see in this one subsystem, but if you don't solve them there, it is going to affect everybody that uses that element of the software and that component of the hardware.

Paul: That's right.

Suzanne: There is a lot. Obviously, you and I share this passion of trying to help these two communities come together. What effect do you think that the work we have done in the national software agenda has on that? Because I see some places in that agenda that immediately say, *Hey, this is something we need to talk to the systems engineering community about.* Has that played into Vision 2035 and things like that?

Paul: Yes. It has really helped inform me a little bit as a participant in the Vision 35 work because I have tried to stay close to all the work we were doing on the national agenda for software research and development. That was an effort to look 5 to 10 years out at what is coming in software so that we can be prepared to help DoD and our other customers and clients. One of our jobs as an FFRDC is not only to solve the problems that people are seeing today, but also to get ahead of the problems that are going to come in 5 to 10 years. There, as you alluded to earlier in this conversation, this move to smarter and smarter systems just keeps coming up. Sometimes, I think people think we are either going to have an AI system or a non-AI system. But in fact, AI and ML, machine learning, are creeping into the systems throughout. Lots of times, like our cars, it is not like we are going to have a self-driving car right away, but we already have AI elements in our cars today that help us with making sure we don't go out of our lanes and help us, maybe, park our cars.

Suzanne: Yes, that annoying beep.

Paul: One thing that I need, because I one time was in a hurry and backed into my garage door because I forgot to put my garage door up. Now they have things that keep you from doing that. They will keep you from backing into your garage door, very helpful for some of us.

Suzanne: And save you some money too.

Paul: Yes. So what we are seeing all along is the introduction of elements of AI into our lives. A lot of people right now say, *Oh, I'm afraid of AI.* But they use AI every day. Every time they go into Google, every time they shop at Amazon, every time they get in their car, they are using elements of AI in their daily lives.

Suzanne: I taught AI as a course in 1989. In that time frame, it was a very limited view as to what we could do with AI and just the compute part of it, all kinds of limitations. Even though people could get the concept, the implementation was really, really challenging. That shift to where neural networks and parallel computing and other things in the computing industry that have made the implementation of AI and ML possible, that is another area where I think there is a place for systems engineering to learn more about, *What can you really do with this?* In many of the systems that we work in, the systems engineers have built systems like this. They built cyber-physical systems in the past, but without that extra boost that you can get from some of the AI/ML technologies. That is another opportunity for learning, is, where can you really leverage that? And, because we have a cyber component, how can you leverage it safely and securely so that you don't end up creating vulnerabilities in the way that the software is interacting with the hardware? That is actually one of the vulnerability points that we have to watch for.

Paul: Sure.

Suzanne: I see lots and lots of opportunities for this. We now have a new directorate on AI and ML. Getting these communities to understand some of that future is going to be one of the key things that we can contribute, and that you can contribute in things like the Vision 2035 effort, to ensure that that doesn't get lost.

Paul: One of the things that actually is challenging and changing, both system engineering and software engineering, is AI engineering, with its strong dependence on data. Now data almost becomes the pre-eminent thing. The data that you use to train the systems, the data that you use to evolve the systems. Learning how to make a good data set for training is going to be a new skill. How do you make sure the data spans the dimensions of the study problem you are trying to solve? How do you make sure the data doesn't have intentional or unintentional bias in it that would lead you to the wrong answers in certain conditions?

Suzanne: I would say that is one of the things that we will be doing in the future, to help these engineering disciplines, is to help understand the impact of data, the impact of hardware/software interfaces to remove and reduce vulnerability. What else do you see the SEI doing in the future to help these two disciplines keep pace with the rapidity of technological change? We are not just talking AI/ML, we are talking quantum computing, biological cellular computing. There is a lot of stuff on the horizon. What are we going to be doing for that?

Paul: Well, I think the work that has been ongoing with DevOps, which leads to smaller incremental changes, but very rapidly, helps tame this exponential growth in science and technology that keeps coming. It gives us a better chance. In the past, when you try to plan, maybe for a system of 5 or 10 years out, it is very hard to do that planning in a world that changes so fast. When you bring that down to the level of a sprint, that we might say in an Agile world, or a quick development cycle in the DevOps world. You are talking now of a week, two weeks, maybe a month or so, it is much easier to tame the future because you can see that far. So I think one of the things that we can bring to the table is this ability to live in a world that is changing very fast by bringing our vision near term and taking advantage of what we know, even as we plan for future evolution in the future cycles. I think that is really a hallmark, anything that exponentiates, whether it is interest in your bank account or software capabilities, it's very important not to start from scratch each time, but rather to build upon the past. If you can keep building upon the past, you can achieve so much, as we have seen.

Suzanne: That kind of leads into one of the other areas that we emphasize is, how does in the DoD space, how does all of this affect the acquisition lifecycle and the people that are doing the purchasing and buying of systems like this that have traditionally been very, very long periods of time?

Paul: I spent my life in that SuZ!

Suzanne: We may have the technical capability of doing quick turnaround, but we have got to also learn to affect that acquisition cycle so that it can accept change in a very rapid kind of cadence. That is one of the areas that I have worked in. I see a big change in that over the last 10 years, but it takes a long time to change the social aspect of the system.

Paul: Well, we have certainly seen a lot of interest in that. In the last few administrations, we have seen people on both sides of the aisle, pushing to reduce cycle time, to open up the experimental window, prototype window, to bring modern software engineering techniques into the fray. I think what they are finding is that it is harder to accelerate the hardware cycle than it is to accelerate the software cycle. One of the things to do is to build the hardware platforms that can accept rapid software changes, so that you can upgrade the capabilities of systems fast. Now, in the commercial world, we see that in some of the software, primarily systems like Netflix and Google and such. They have done a real good job in that. But we also started to see that in some of the hardware-intensive systems like a Tesla car, where they can do upgrades overnight in the car. The Air Force, following Tesla's lead a little bit, recently updated software on aircraft in flight, while it's in flight, not when it's on the ground getting its maintenance, but in flight, gave it a new software load. I think we are going to see more of that; built platforms that have certain physical capabilities, but where we can change the software much quicker, so that we can continue to bring new capabilities to soldiers, sailors, airmen, marines, and now, we have to say guardians too.

Suzanne: Yes. Space Force.

Paul: Yes.

Suzanne: I want to thank you so much for talking with us today. I always enjoy these conversations with you. This topic when it was brought up, I was like, *Oh yeah, I want to do this one.* I think, like I said, you and I have this passion for looking at these interfaces between these communities and trying to make systems engineering and software engineering more effective together. [It's] what is going to get us the capabilities for quick turnaround for things like, scarily enough, software update in flight! That is one of the things we have been talking about for decades. And it's like, *My gosh, it's finally happening.* You know that is part of the world we live in, is how do we do these things? I really appreciate your insights. I appreciate very much your work with the Vision 35 group, because being represented there at that table is actually one of the things that is a sign to me that we are really getting much more synergy between our disciplines. I look forward to the work that comes out of that.

Paul: Well, I am excited about the future. I think engineers should all be excited about the future. When you bring different disciplines together, that is usually a time of really fertile innovation. I think we are getting to that point where we can do that. I want to thank you for this nice interview, SuZ, and take care of yourself, OK.

Suzanne: Absolutely. We all have to take care of each other right now. I do want to make sure our viewers know that we'll include links in the transcripts to things like the Vision 35 project, and other things that we've mentioned, model-based engineering, and such. They will be able to see that through the transcript. I want to thank you, and I want to thank our viewers for joining us. I think everyone's going to enjoy this podcast when it comes out, and we are going to go off and create the future.

Paul: You bet. That is what we do. That is what engineers do.