Challenges and Metrics in Digital Engineering

Featuring Bill Nichols 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 joined by my friend and colleague, Bill Nichols. Bill is a senior member of the technical staff in the SEI Software Solution Division also, and he has been a frequent guest on our show to talk about many types of cool work he did with us here at the SEI. Today, we are going to be talking about his work in digital engineering adoption. Welcome, Bill.

Bill Nichols: Thank you.

Suzanne: For those that haven't heard from you before, tell us a little bit about yourself, what made you come to the SEI, and what is the work that … I would like people to understand the scope and breadth of the work that you do here because you do a lot of different things.

Bill: My background is in physics. I did my graduate work at Fermilab some time ago. But a lot of the work in physics today is doing things like data analysis and data collection. So that was kind of a natural segue into the work I did industrially in nuclear engineering, where I worked on scientific and engineering codes for use in nuclear-reactor design. Around the turn of the century, we started a project at our lab to update a lot of the older codes, many of them dated back to the 1960s. Literally the first Fortran compiler outside of IBM was used on some of these codes. So they were really aging, and they needed to be upgraded, a fairly large body of code needed to be updated. It was between a couple of laboratories. I led a project that took the lead on going through this effort. That was my first experience with software engineering at some scale with cross-disciplinary teams. We had engineers. We had programmers. Of course, we had to do management reports. So that really got me into the program- and project-management aspects of software development. Now, the SEI had come over to help us with some of that. That is where I met Watts Humphrey and some of the others.

Around 2006, I decided I wanted to see how some of this was being done in the rest of the world. I left the nuclear lab and came to the Software Engineering Institute. For many years, I did coaching on software development and software project management, and that really got me deep into the metrics of software: How do you measure and monitor software progress? How do you track the projects? What should you be measuring about productivity? So most of the work I have been doing since then has been on software metrics and measuring progress and productivity for software development.

Suzanne: A lot of your background is measurement related, and you are very well known in that space. And now we're talking about digital engineering. How did you get into the digital engineering space?

Bill: Well, we actually did a lot of digital engineering type of activities, even back around the turn of the century at the lab. That is where we started doing things with round-trip engineering tools. Digital engineering is, to a large extent, what we were trying to do on my project, which is to create reusable artifacts that would live over time. Now, in this particular project we were working on, where I gained a lot of this experience, I was working with the folks who developed and were implementing ACVIP [Architecture-Centric Virtual Integration Process], which is the architectural virtual integration. We were doing a project for Future Vertical Lift to see how the introduction of the digital engineering techniques would affect the program. In particular, they were looking for things like, *Does this help with productivity? Does it help with quality, communication?* Those sorts of things. That was the genesis of this particular project where we got a lot of this experience.

Suzanne: You know me, I am all about understanding the sociotechnical aspects of things. When I read your blog post on this topic, which we will reference in the transcript, of course, I did really note that many of the challenges were more on the social aspects than on the technical aspects. There definitely are technical challenges. But digital engineering as an idea, as you say, has been around a long time. It first got introduced to the DoD in a large scale through the National Defense Authorization Act in, I believe it was either 2017 or 2018. That is when everybody went, *Ah, we have to do digital engineering*.

One of the things that I am seeing in the places that I work, and I want to see if this is the same thing in terms of your notes of what has been challenging and also noteworthy about digital engineering, is the idea of trusting the models that come out of digital engineering. Because if we want the digital engineering models—and I am using models in the very broad sense—to be used over time—because that's one of the big selling points, is we can use these models over time, long sustainment tails. We have got, and what we call an authoritative source of truth, also referred to as ASoT, in the model, but I don't have an authoritative source if nobody trusts it. That is one of the things that I am really interested in is, what did you find as either … Did you see that same challenge? And if you did, what were some of the things that you have seen to be successful in helping people to build trust with these digital models?

Bill: Well, there seems to be a lot of work to go. Part of the issue that we were coming across was there are multiple levels in doing this sort of engineering. You have to deal with things like the requirements. You have the architectural-level design. You have code generation. You have actual code instantiations. Each of these levels, you have to have a model that is reviewable and has some sort of authority. You want a single source.

Now, if you are going to make that work across multiple levels, the models have to be consistent with each other. That is, you have to be able to peel the layers of this onion. One of the issues that we were coming across is the different models didn't really fit together well, and that had a lot of implications. One of them was that you tended to have modelers at one level doing work that was meaningful to them, but it was difficult to hand off to another group because they didn't really have the same level of experience. When they got done making their changes, there was no backward path to keep all of the models consistent. That really undermined some of the authoritative source of truth. It made for a lot of rework.

Part of the reason you want this authoritative source of truth is you want one place to go to. It is a fundamental design principle that you want to remove redundancy. That is, you don't want to express the same thing in multiple places because once you start doing that, they are going to get out of sync. Then you are not going to know which one of these is the right one. So that is one of the problems that we started to run into, and that was a lack of the tools having this two-way flow. That has some other implications, by the way, especially when you get into things like modern development. You want to be able to have feedback loops. You want to be able to do some development, get some experience over what are the strengths and weaknesses of this. It is certainly important when you are prototyping. And if you can't do that round-trip engineering, you can't really shorten the verification and validation loops.

Suzanne: That is one of the things that I know in all of the SEI's modeling work that we have been doing is emphasizing this idea of virtual integration, virtual testing, and early validation so we don't have all of the implementation decisions in concrete when we find out that, *Oops, this assumption we made in the architecture turned out not to be correct*. That, to me, has always been one of the great benefits of model-based engineering in general and digital engineering tools. What are the biggest challenges in getting that piece of it, getting that validation and verification and integration piece understood by all the different layers of the stakeholders so that they participate actively toward getting that authoritative source of truth? Have you had any success with that. Because I tell you, that is a challenge that I have in the programs I work with?

Bill: It certainly is a challenge. The first thing that we noticed was, if you are making these sorts of changes in digital engineering and your objective is to shorten these development cycles, you have to actually change the workflow and build that into the cycles. You can't just throw digital engineering at the problem and try to do things the same way. You are fundamentally changing the workflow. That means you have to change the workflow. You have to change the way you are tracking the workflow. You need to make sure that people are trained in this new approach, and you have to have some way of tracking the work.

Those were among the biggest problems that we encountered, that they were trying to do the work in ways that weren't so different than what they had done. That is probably to be expected. I think part of the issue here was they were trying to implement a lot of changes at once. And these big-bang changes make it very hard to really become expert in any one of them. If you don't gain that experience and the expertise in using a single aspect of the tooling, then kind of nothing works, and you always revert back to what you know how to do.

Suzanne: Yes, and we talked about that in change management as having a period of integration and practice being needed as one of the foundations of having an adoption of anything be accepted and to stick. I work in environments as well where it may not be necessarily multiple digital engineering tools, but it is multiple change initiatives at once that blows people's heads apart.

But in terms of that integration in practice, I find that we have people that become very expert at, say, Python scripts for aggregating a bunch of data and presenting it. And we have people that get very good at using DOORS in a way that actually lets you see the requirements model. And then there is the AADL [Architecture Analysis and Design Language] and all of the attributes. To me, this sends you to the idea that you are going to need a team of model-based engineers that all work together on getting to this. Have you found that to be a teaming construct that you can get a program to actually accept? That it is not just one modeling expert for the whole program, but that we need these different kinds of modelers working together to build the set of authoritative sources of truth that are going to take the program forward.

Bill: I will take that a step further. And that is that there was a tendency to do the modeling that you needed at a specific level without a whole lot of thought about how the overall model would fit into this family of models. There was probably a lack of mentoring available to help the people doing the models understand how best to use them and how to integrate the information from the other models and to communicate those models to the handoffs. We saw problems with the handoffs and a lack of expertise on the specific tools and applications. I think there is a lot of environmental preparation that could have been done differently. We would get a more fair evaluation if we get a look at it.

Suzanne: I like your concept of a digital modeling environment as being the home for all of these people, families of products, and artifacts that we want to move forward and be able to use in sustainment, not just in initial development. This goes back to another concept that we have at the SEI of evolutionary architecture, evolvable architectures being a key. I see that as being a key to digital engineering models as well is, if I create an architecture that is so closed that when I learn things, I have to go back to the beginning to be able to evolve it. I can't just evolve a piece of it that is under scrutiny, under test, then I have a much bigger job, then I am less likely to do that work. I just need to get it out to the next step, which goes back to your workflow idea that I have to change my workflow so that I can get a workflow that explicitly includes, *Hey, how does this affect the model?* And *What are we going to do to make sure that the model evolves consistent with our current implementation?*

Bill: Along with that, there is another problem. That is, doing the design at each of these levels at the right level of abstraction. One of the problems we observed was that there was really a tendency to try to use these tools and sometimes get a little too much into the weeds for the specific type of application we are using. Think about the different types of tools and models that you are creating. You are going to be doing some requirements work. As you start getting into architecture, you might be using some SysML. As you go into more detailed modeling, you might be using AADL, which is really getting into the component-level development. But you don't want to be doing that kind of information at a higher level like SysML. And you certainly don't want the SysML type of representations, where you are going to have the start of component breakdowns, sequence diagrams. You don't want to get into that in the business notation that you are going to be using for the requirements.

Suzanne: Yes, I see exactly where you are going. It goes back to that family of models and family of modelers as a way of making sure that we are doing the right things at the right time.

Bill: Exactly. One of the problems that you do have that makes this more complicated in a DoD type of environment is you usually have multiple organizations involved. There are implementers. There are integrators. Sometimes they are the same organization. Often, they are different. Often, they are completely different companies. Sometimes they are contractors. Sometimes you are dealing with government. So you really have to be conscious of what are the handoffs. What is the proper level of abstraction for these handoffs, and are the representations going to be sufficiently precise that the people you give these to are going to be able to interpret them without ambiguity?

Suzanne: That leads to another connection that people don't often think about in terms of digital engineering, which is the acquisition of a program's artifacts using digital engineering because now, as you said, if I am handing off from one company, one organization to another, I may have intellectual-property issues and data-rights issues that are not things that engineers think about, but the management of those companies are very conscious of. And so as an acquisition agent in the government, we need to be aware if we are planning to use digital models as a source of truth going forward, that we have the right things available to be handed off, because there could be…I have seen cases all the same that way where everything that you need is in the model that is being handed off, but not everything that is in the model is handed off because of data-rights issues. You want to deal with those issues up front. Have you seen that effect? Because I have had to experience that. I don't know if the projects that you've worked on did a better job of getting that taken care of in the beginning than we did.

Bill: Well, we didn't see some of those things specifically, but it is real easy to see how those kinds of issues can creep up. One of the things we did observe over time, the program did become better at using the digital artifacts for things like reviews. Among places you get into trouble in the real world are, you do all of these technical reviews, and you have essentially created PowerPoints. You are kind of defeating the purpose if you take a Cameo model, take screenshots and throw it into a PowerPoint for your review.

Suzanne: Yes, that is a signal that you haven't really adopted the digital engineering mindset, I would say. But it also reflects the fact that the engineers may have adopted it, but the management hasn't. And so …

Bill: Or that they haven't figured out how to actually conduct the reviews. One of the aspects of the digital engineering is, it is not just a matter of training the engineers. These reviews are for typically management and non-technical people, and they have to have an entirely different way of interfacing with this information. So you need to be able to explain that to them with things like the Cameo model and show them at the appropriate level of abstraction what it is they are looking at.

Suzanne: One of the things that relates to that is you have got different skill bases in your acquisition and management teams, both from the digital engineering engineers per se but also amongst themselves. I know some acquisition agents that are very comfortable diving into a model and navigating around. You just basically give them five minutes, and they are gone. Now those folks can cause challenges because they end up looking at things that may not be done yet. If you are giving them a live model, they may end up being a little bit disruptive in terms of the work. So there is that side. Then you have the other side, of the managers and acquisition agents that have no experience using these kinds of models to gain information that they need. And they need to essentially have their hand held through the whole thing and ask their questions and kind of have a toolsmith with them to say, *OK, this is where you would find that*. Those sorts of things are of the challenges that we have run into, and we basically have assigned toolsmiths for review types of activities so that anybody that isn't comfortable navigating the model can have somebody that will help them find things. Did you use any of those kinds of techniques to get people past that?

Bill: Well, that was beyond our purview. We were really there to do the evaluation and measure what was actually going on. But among the things I did look at was, I followed up some of this with a survey. We found that a fairly substantial portion of the people involved didn't really feel like they had adequate understanding or training of the digital engineering environment. That is really understandable when you consider all the different levels of work that had to be done and types of handoffs that had to occur. So again, that gets back to the idea of preparation, understanding what the workflow is, and having the staff properly trained to use these tools.

Suzanne: So those are all factors, and you talked about all of these in your blog post. I want to move over, if you are ready, to the measurement aspect, which is one of your areas of expertise, because the thing that people want to know is, is this better than the old way? I mean, that is a fundamental thing they want to know now. There is a lot that goes into answering that question. But what are some of the measures that you have found useful in understanding whether we are actually improving our ability to use these models and if the use of these models is actually improving the success of the projects?

Bill: Well, that is actually a much harder thing to show than you would think, for a variety of reasons. The first of which is, what is your baseline? One of the things I did was I looked back at the SRDR data and looked at what were the actual program costs on similar types of projects. When you try to normalize these by some sort of scale, I found that the range, the variation in the range of productivity was so large that you can't really pin it down. So with a one-of, you don't really get a sense of how this fits into this broad perspective. What is the productivity?

But there is a deeper problem. And that is, the way we have been measuring projects more recently does not really shed light on some of the important aspects we would like to see. You want to be able to do things like demonstrate you are finding issues earlier. That would be one way of doing things. But that also means that you have to have your process instrumented in a way that you are actually tracking the rework or the issues. But many of the projects doing this sort of work don't really put issues as such into the system until they have gone through system test. So if you don't have this information in process, you don't really know. You don't have insight into those feedback loops.

Another problem we came across is that you have this notion of doing things iteratively and incrementally. So you have this item of work on implementing some kind of feature. As it goes through the development stage, now you want to understand, *Where am I applying effort on this?* You expect in a digital engineering world, you are probably doing more modeling. You should probably be doing more design work up front, and you would hope to see less rework at the back end. That turns out to be relatively hard to get unless you have explicit ways of isolating rework. The way they were setting up typically things like storyboards would be, you have a story for the implementation. You have a story for the modeling. You have a story for the virtual integration. But unless you have some way of tracing that entire thread through each of these individual stories, you lose visibility into how each of these are done.

You also have to measure how much work did you actually put in the story. They were giving us the total effort applied in a sprint. And they told us how many stories were planned, how many were done. Well, how many of these were virtual integration versus virtual analysis? Well, we didn't really get that. And it didn't matter because they weren't necessarily the analysis stories for the same virtual-integration stories versus the same model development. So you lost a lot of that integration. You lost a lot of that detailed information you would like to see to understand how this is really working. What we fell back on, given the limits of this particular program, were looking at some measures of where the effort was applied. We were finding that they appeared to be putting more effort into the program early, and they did appear to be going through the final stages more quickly. Now, unless you actually measure direct outcomes, though, that is hard to get because unless you measure individual story by story, it turns out the actual measure you get is just their estimate with some windage factor based on the velocity. It makes things rather difficult.

Suzanne: The other thing that I am seeing that is related to that is in terms of using the tooling that would allow you to do that, for example, JIRA. I know engineers that only update their JIRA tickets once a week. You might be able to get them to update it once a day, but you don't really, in terms of the kind of time you are looking for, how much calendar time, how much effort, you can get the effort no matter when you input it, but you don't get the calendar time if you are not actually using those tools to say, *OK, I finished this job, it's going into its next workflow state*. And we run into that problem a lot in terms of … There are lots of problems in measuring these kinds of things. But I want to come back to the idea …

Bill: Just a follow-up comment on all of that. I think what we were seeing was actually an artifact of, the way they have set up their measurement was for an entirely different purpose than measuring the effectiveness of the process. What they were trying to do was measure progress to completion, for which the tracking was actually reasonably effective, but you needed a different type of measurement if you wanted to start to understand insight into the effects of process improvement. And changing their measurement system, once they realized that what they had wasn't really up to the task, was too heavy a lift.

Suzanne: That is what I wanted to get at is, many of these measurement systems for progress against completion are put together early in an acquisition with assumptions about how the workflow works. As soon as we change the workflow, then we have violated the assumptions of that measurement process, and then you are left where we were talking about where the measures that are being collected are not the ones that are effective at giving us answers to the question, *Are we more successful doing it this way?* I am kind of harping on this because those of you that are in the acquisition community, some of this you are going to have to deal with. You may say, *Oh, I am not an engineer*. But you affect the ability of digital engineering to be successful on your program. Sorry, I am on the soapbox.

Bill: Yes. Well, let me give you a couple of observations on that. Clearly, one of the things that has not been agile is how we instrument and measure our projects. That is about as truly waterfall-ish as anything in software engineering has ever been. Once it is set up, it is as though it is cast in concrete. That is a problem because the way we do the work is constantly changing. The assumptions, as you said, the assumptions under which a measurement program was instituted, if those assumptions are no longer valid, then you aren't getting useful measures anymore. One of the big problems with measurement is you get what you asked for but not what you need. You really have to take the measurements that you think you need, do some prototyping, do some modeling or simulation, and make sure that you are getting the information out of them that you think you are getting out of them because often you are not going to.

It is a source of miscommunication. To quote George Bernard Shaw, the single biggest problem with communication is the illusion that it has happened. Throwing those CDRLs over the wall doesn't mean you are going to get what you need, but you probably will get what you asked for, at least what they think you asked for.

Suzanne: Yes, we see that a lot and not just in digital engineering. This goes back to the, *We want these models to be long-lived. We want them to satisfy our needs for development*, and *We want them to satisfy our needs for sustainment*. If the measures that we are using are not in tune to both of those aspects …

Progress toward completion has a very different meaning in the first minimum viable capability versus the long-term evolution of that capability. You really have got two measurement systems just out of the get-go if this is a long-lived system.

Even beyond that, as you said, the measurement system we set up when we were conceiving the program, maybe not even thinking about digital engineering versus our implementation with digital engineering. Those can be really out of whack.

Bill: Yes. That is why one of the things we recommend doing is trying to track the rework. In particular, when you do have these issues occur, do those real deep dives. Do the root-cause-analyses types of work so that you can understand what was actually effective and what wasn't.

You can get a lot more information out of that than you can out of a lot of other measures. What were the sources of the rework? What did you actually do wrong? What did you have to change?

Suzanne: Yes. You know I am a big fan of root-cause analysis, even though it is some work. I think the other thing is that we get so hung up on quantitative measures, I will call them. The root-cause analysis is really a much more qualitative process, but it yields very rich information. So we have got to be sure that we engage with both sides of the measurement equation, both the quantitative and the qualitative.

Bill: That is what I would call triangulating. You want to make sure that you do the root-cause analysis here so that you understand qualitatively what actually happened. But you want the quantitative information, so that you can do some *what-if* evaluations. We did a couple of things in this program to try to get a handle on the potential improvements from the digital engineering. One of them is we actually asked the implementers to go back and give us their best estimates on what this would look like if you knew then what you know now. That is, give us your best estimate of where you would have put this effort and how things would have happened if you had been smarter the first time. We found that we got some pretty consistent results suggesting that, yes, they realized if they had understood the lessons they learned before they started, and they had a do-over, that they would definitely do a better job, and you would see a more pronounced improvement.

Suzanne: That is telling because that goes back to your point about training. The kinds of lessons that you have got in the blog post, those are things that lend themselves towards mechanisms like training. So looking at these early results from projects like Vertical Lift and others, I think, is one of the ways that we help the community to build more confidence in digital engineering and build more capable engineers and acquisition agents to be able to take advantage of it. Because that is the end goal, is for us to have a community that can take advantage of this for both our development and our sustainment.

Bill: We did one other thing, by the way. That was at the end of the demonstration, the implementers were asked to add some new features, we called them excursions. We tried to do some additional measurement on the effort required for these excursions. One of the outcomes of that was, although the measurement paradigms hadn't really changed, they were still using the same approaches, they had a better idea of what they were doing. The products, the final products, actually did appear to be more easily modified and implemented.

Suzanne: Excellent.

Bill: You can only draw limited conclusions from this sort of thing because it was a one-off, it was very qualitative. But there were definitely some encouraging signs that we are moving in the right direction, we just have to get smarter about how we do this.

Suzanne: So the big challenge in our work is, you have got some encouraging insights that, from a research viewpoint, you would need to do a more formalized study, have a more controlled environment, where you are confident that people are changing workflow states, different measurement programs, etc. But we mostly work in real-world projects. They have already got an acquisition in place. They have already got a measurement system in place. However they are doing the modeling is however they are doing the modeling. So we don't have influence over all the factors that we would include in a research project. So my question to you is, how are you working through that? Because I know Bill Nichols. I know there is a plan in there for how do we get to that more robust understanding of this. Are you working that?

Bill: We are working that. Let's start off with … The fundamental problem in doing software engineering projects is you can't do controlled experiments, at least not beyond the trivial. You can find a few experiments where you can have multiple teams doing the same problem, but you just can't afford to do these programs multiple times with different approaches. You are basically stuck doing this with observational data, with all of the risks that entails. Basically the kinds of approaches that we are trying to encourage is to think about what are the things that you really need to measure and adjusting your measurement programs so that you actually are getting some of this.

The biggest thing we thought was missing was the different ways of measuring quality, things like tracking the issues of the defects at an earlier stage, trying to demonstrate that you are actually finding issues in virtual integration or virtual analysis and not having these come up in the physical construction. That is something that actually should be fairly straightforward. We have some pretty good ideas of what kind of problems have come up before. We also know from information going back to, oh Barry Boehm's TRW days, that if these issues don't become visible until you are doing, say, physical testing on components, they could be easily hundreds of times more expensive than fixing them in the modeling stage.

Now you have got a couple of things that you can actually look at. How much effort is involved in rework when you get into real testing? And how many issues are you finding in virtual integration? Are you actually finding anything? If you aren't finding it, maybe it is because your tools are so good that you are not introducing them. But that should also mean you are not going to find the issues when you get into the real world. If you are finding the issues, now you have some other ways just by how much time you spent in this entire activity, you have at least some handle on how much effort fixing those things took. So you want to see that this early rework is indeed less expensive. We did a paper at the Acquisition Research Symposium and we showed the traditional V. What you do is you try to basically collapse the V. You want to add that additional V. It's like an inner V where you are doing a validation before you get to the big test. You are doing that validation digitally.

Suzanne: Yes, and that allows you to do incremental and iterative as well.

Bill: In principle, that should allow you to do iterative development in a way that the feedback from a product that has already been designed and constructed and tested, you couldn't do. You want to make those course corrections early.

Suzanne: We have a wide array of things on this topic, which you can tell that I love this topic. I always want to put myself in the position of somebody who is hearing this for the first time and is going, *Oh, my God, I've got to do something about this*. If I am on a team or managing a team that I need to adopt digital engineering or I am at the beginning of adopting digital engineering, what are the kinds of resources that you would recommend? What are some of the things that the SEI is working on that they should look for that we can help them with?

Bill: One of the things I would definitely do is go back to, I think there were a lot of good links in the blog post. You can look at things like how AADL has been used. There is some good information on digital engineering. On authoritative source of truth, Adventium, who was one of the partners on this program, put out a couple of very good papers [e.g., this one on authoritative source of truth], as did the Object Management Group [OMG]. If you go back and check the blogs and do a good search on the SEI website, I think you'll find places like OMG, Adventium, and various other links are going to provide a lot of information. My advice would be if you are going to go into this sort of thing, go and use the Change Management 101 and try to keep the scope limited, try to keep the changes limited at any one time. We tried to change a lot of different things, and it also made it very hard to tell which ones were actually being effective.

Suzanne: Yes. I have run into that one many times. We are going to do all these things and tell us how much this one improved our efforts. OK. So what about you? What is next for you? Are you going to stay with this work? Are you going to do something else that's exciting? You know, what else do we have to look forward to from Bill Nichols?

Bill: Well, the amount of effort we have for this might be limited. I am certainly working on this certainly on the backburner and providing some help. But I have been looking more carefully at the DevSecOps instrumentation and how to get more information out of the DevSecOps pipeline for other purposes. The big one I am concerned about right now is, *What kind of programmatic information can we give to the senior managers based on reliable DevSecOps data?*

Suzanne: Excellent. Well, that will be useful for a lot of my acquisition clients, I can tell you that.

Bill: Well, DevSecOps has put process back on the map.

Suzanne: Has it not? Yes, it has indeed.

Bill: It's made process respectable again.

Suzanne: Although I do find people like to talk about workflow more than process. It's kind of funny how that …

Bill: Oh, yes, I assure you. It's a totally, totally different world.

Suzanne: Yes, exactly. Exactly. All right, well, I want to thank you for talking with us today. It is always enjoyable having a conversation with you. This is a topic that we are both engaged in from different aspects. I hope that the resources that we have provided, the things that we have talked about will be mentioned in the transcript, any of those resources from Adventium and others.

Finally, a reminder to our audience, our podcasts are available just about everywhere—SoundCloud, Stitcher, Apple podcasts, Google podcasts, as well as the SEI's YouTube channel. If you like what you see, please feel free to give us a thumbs-up. I look forward to talking with the next guests or talking with Bill again about new things that are happening here at the SEI. Thank you for watching.