SEI Fellows Series: Peter Feiler

featuring Peter Feiler as Interviewed by Will Hayes

Will Hayes: 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 United States Department of Defense and operated by Carnegie Mellon University. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

My name is Will Hayes, and I am a principal engineer here at the SEI. Today, I am pleased to introduce you to Dr. Peter Feiler, technical lead and author of the Architecture Analysis and Design Language, more commonly known as AADL. Dr. Feiler will be the second researcher featured in our SEI Fellows series.

These researchers are so named because of their outstanding contribution to the work of the SEI, and from whom the SEI leadership may expect valuable advice for continued success in the institute's mission. He has lent his expertise in this area to several recent Department of Defense projects, including the Army led Future Vertical Lift program, which will bring in the next generation of military helicopters.

Today, we are going to talk with Peter, who was named and SEI fellow in August of 2016, about his career, his work on AADL, and his thoughts on the future of software engineering, and more importantly securing safety-critical systems.

Welcome, Peter.

Dr. Peter Feiler: Thanks for having me.

Will Hayes: It is not often that we get to speak with someone who can bear witness on the evolution of software engineering as a discipline, and who can legitimately be credited with substantial contribution in a new field, particularly in model-based engineering.

I like to think of advances in programming languages as providing more structure and more help to the people who create products that eventually get converted to zeros and ones.

But the work that you have done has far surpassed the complexity and aid that compilers provide. The modeling language you have developed and the work you have focused on in your career has really helped to assure performance and correctness of software in very profound ways. So where do we start?

Peter: Well, first of all, thanks for the compliment. I will go quite a bit back. One reason is because all of those pieces in my history played together and came together when I had to do the work in AADL.

I started studying in Munich in 1971. Two of the professors I studied under was Prof. [Richard] Baumann and Prof. [Friedrich] Bauer. Prof. Bauer was involved in coming up with Algol as a language and that the first language I learned how to program was ALGOL 68, which was a relatively high-level language at that point.

In 1974, I then had a one-year scholarship, which led me to Carnegie Mellon. That is where I came across Nico Habermann, who became an advisor. His history is interesting too because he got his Ph.D. under Dykstra. At that time they were doing the THE operating system, which pursued the concept of virtual machine layering.

Nico then crossed paths with David Parnas at CMU in 1967. That is where both of them got the idea on how to combine the module concept with machine-layering concept. So when I got to Carnegie Mellon, Nico was doing this project called family of operating systems where we architected an operating system in such a way that combines those two concepts.

Later on, I then took this concept and brought it to the CM* project, which is a multiprocessor project led by Anita Jones. So while Nico Habermann was on sabbatical, I continued with that work. On the side Nico Habermann's thesis work itself was on the use of semaphores and how to avoid deadlock. And that was actually one of my interest was dealing with concurrency and all that, so I continued working with Nico on the side on the topic and notions of past expressions.

During my time at CMU, I also took a seminar with Mary Shaw on language design. I finally did my thesis work on a project called Gandalf, which is generation of software development environments from a specification of languages.

Will: So if you could speak in perhaps layperson's terms about what Gandalf did, because I have heard a little bit about the background on this, and it seems a novel introduction of concepts at that time?

Peter: Yes, at that time what happened is people traditionally were writing code in a text file and then handed it to the compiler, and then it ran. What we said *is Let's create an environment*

where you interactively are doing these things, and the system knows the language and it acts as if it was interpreting the code. So at any point in time we were totally up to date.

Today we are familiar with those kinds of environments. The first one that was pursuing that idea commercially was actually Rational with their Ada environment. Later on you find it nowadays for a lot of environments and this was in 1979. The way we approached it is to say, *Can we do that work by actually generating it?*

It is like partial generator, just generalize to generating complete environments. Again those concepts you can find today in tool environments like eclipse. There is the eclipse modeling framework. Now this X text generation environment technologies that we use to implement AADL.

Will: And people take this for granted.

Peter: Today they did not take it for granted. We did that as thesis work, so I could reach back to what I did for my thesis work when I had to implement the AADL standard. That is kind of why I am telling you the history. A number of those pieces all are relevant to the kind of issues we are trying to address with AADL language.

So when I was done with that I went to see Siemens. You have that listed. We started the research group and the software research group in Princeton at their corporate lab there. During that time we also had a chance to spend a year out in Oregon where we had 200 Siemens people doing work with 200 Intel people on a new processor architecture that became the i960 processor.

Then when the SEI opened up in '85 I came back to the SEI. [I] ran across people like Mary Shaw and did some studies on user interface, got into configuration management, [software development] environment work, and became a program manager. Did that for six years. After I stepped back from that, I got involved in some of the work Lui Sha was doing with Simplex.

Bruce Lewis then got me involved in the AADL work. I have been working with Bruce since 1997 under his funding to turn some research work into the standard and moving forward where we are today.

Will: The sophistication, the process of growing more and more sophisticated in the way we manage the production of that long string of zeros and ones, really the formalization of that is really where your career is focused?

Peter: Yes, and always has been. I have always left in notations that have strong semantic, strong typing that as early as possible we can identify issues that you may have in your system.

You want to catch things when you introduce them. You not want to drive a car and find out when you go down the steep hill that your brake does not work.

Will: Looking at the influence of this kind of work on the products and services we have come to take for granted today, the level of discipline and foundational work that underlies the techniques in use today, I think most folks do not really realize how much careful work went into it. You are one of the people who was there making it happen from the onset.

Peter: Well, and taking advantage of other work that has happened, because some of that stuff, for example, came out of Xerox PARC . Being at CMU I was already working on machines that today are known as Macs, things with mice. In 1976 I already worked on machines with bitmap display menus and mice and stuff like that. I have had very early exposure to some of those technologies, which then helped me pull off a lot of that stuff together when we worked on the AADL standards. It wasn't just language design, but everything else around it—taking it from a programming-language-level—like ADA was very good for safety critical systems, because of it having strong typing and all that. We took over all those principles and just moved it into the architectural level as well. At the same time, the importance of architecture has been pushed by the SEI through some of the other work in our architecture group as well.

Will: Just as strong typing influenced programming languages to be more reliable, more secure, the concept of typing applied in an architectural sense seems to be where AADL is going.

Peter: That is one element to make sure that what you specify in this notation is consistent and fits together. The other element of it is what we wanted to do is put in semantics into the things that you express in AADL that are close to what we are interested in, namely embedded software. Software by itself does not do anything.

You can talk about the function of software, but if you want to talk, say performance, you only can talk about that in the context of it running on some hardware. If you want to talk about safety, you have to talk about how it runs on the hardware, how good the hardware is in addition to how good the software is, and how well it interacts with the physical environment that it deals with because that is where the problems are.

Industry studies have shown that for software systems that are called embedded systems or cyber-physical systems people call it today, 70 percent of the problems are introduced during requirements and early architecture specification; 80 percent of them are not discovered until post-unit test.

It clearly indicates we know how to write the functions, but it is the playing together that matters, and that is where we did not have good tools. That is what this notation focuses on is *Can we*

define that part of it so that we virtually can integrate a system and virtually already determine these pieces will play together.

Will: So rather than having to wait for the hardware interaction to occur and observe it when it has already been encoded, the abstractions that are available to you allow you to simulate and to really compile and test and run the model prior to that time.

Peter: Yes, exactly. What happened is they had an aircraft sitting there. They had a software build. They put the build on. Nothing happened. There were some configuration issues in the build. So then they fixed those. Then the first time they started up the aircraft with the software, then the screens started going blurry. They had no idea why it was blurring. It is things like that. The thing was still sitting on the ground, so we had not even flown yet. Then it just continues that way.

You kind of explored the thing, and we want to engineer these things. That is what made it hard as we moved forward and put more and more software into these systems, so they were not in isolated pockets anymore. You cannot drive a car anymore without the software running. You cannot watch TV anymore without the software running. You cannot even use a refrigerator anymore without software running.

Will: With the Internet of Things all of those devices you just mentioned speak to each other as well with the software running.

Peter: Exactly. That is what we are facing nowadays. That is why this whole issue of safety-critical and safety and security are just so important. If you do things right, we actually can save lives that way. So even though not being a doctor, I can save lives now.

Will: OK, Doctor Feiler. What are the new frontiers? Where are you pushing it now?

Peter: We have we have come from the safety side because that is the domain we have come from. Both from us within the group and being at the SEI, security has come into the picture. Also, the research community has done that.

There is a DARPA program called HACMS [High Assurance Cyber Military Systems] where they are interested in making these autonomous vehicles secure. One of the teams is Rockwell Collins that has been always very strong in model checking type-of-thing. Rockwell Collins, about eight years ago then embraced combining model checking with AADL to get the scalability (https://www.umsec.umn.edu/sites/www.umsec.umn.edu/files/NFM-springer-proof-72260126.pdf).

So we have been working with them together, so they brought that technology that had its roots in safety and applied it to security (http://loonwerks.com/projects/hacms.html). When they did

that on a UAV [unmanned aerial vehicle], after 18 months they then gave that system to Draper Labs. They had a red team trying to break in from a security perspective had access to the source code. After six weeks they were able to break into it from a security perspective.

Will: Is it like saying not only are we modeling the behavior of the hardware, we are now starting to model the behavior of humans interacting with the system?

Peter: And the software and the assumptions that are being made, because most of the security issues are low-hanging fruit holds. The sophisticated ones people do not even have to bother yet. Buffer overflow type of things are the easy ones that people kind of break in today. Those are the ones that were closed up by the work that these guys were doing.

There is some behavior interaction where I can get you to be out of sync with me. *You think I am doing this, but in reality I am doing that*. They have started addressing that as well. Then the next layer will be *By playing around with time, I can get you get confused by pretending like I am at a different time and so on*. Some people have not had to use those kinds of issues yet, but we see those problems currently from a safety perspective. There are plenty of examples that we can point to our software has cost accidents because of those kinds of things.

Will: So clearly this field is continuing to grow. What do you see as the major milestones coming?

Peter: Going forward, on the security side the approach has been let us build a firewall and protect us from the outside world. For those of you who know Arlo Guthrie, he has [sea] shanty where he says, *We can't build a fort all around America*. We cannot protect ourselves from the outside world. We need to make sure that we are resilient, robust inside, whether it is from a security perspective, breaking in externally, or from a safety perspective, because we never can write zero defect software.

Even unintentional issues still have to be faced. What we need to do is make sure that there is robust kernels that are verified on top of firewall type of mechanisms so that just because I am able to break into your entertainment system on your car, I cannot send commands to the autopilot and the brake system. Those need to be isolated. They need to have authorization to do that kind of stuff. It is those principles that just need to be pushed through.

For a long time people were arguing in embedded systems, *We do not have the cycles to do that*. But even for your desktop, your desktop spends 80 percent of its time virus checking and doing stuff like that, which we would not have to do if you would have strong hardware support for that kind of protection. I think, on that front, is one area where we can make some progress in technology that is a combination of what we do on the hardware side and what we do on the software side.

Will: So an allocation of those concerns to a different layer in the architecture as a structural influence on the industry as a whole.

Peter: Also shifting it to say, *We cannot guarantee things upfront, therefore we need to put some sanity checks inside, so you do continuous monitoring*, which is stuff that we do on the security side that monitor toward the net traffic. Then also put from the beginning protection mechanisms in place so that we can get better isolation.

Will: So instead of virus checking being an application that is installed, virus checking is a presumption that exists in the system when it is first built.

Peter: Also that once you get into a certain area of your system, you are then limited to what you can do at that point. Once I am inside your PC, I can do anything I want. That should not be the case.

If I am in from one window, an app type of interface, then all I should be able to do is access certain things, but that requires enforcement of those things that occur, and currently we are too coarse grain.

Will: So the modeling approach that would evolve from what you are talking about would force those concerns to be addressed prior to anything being built?

Peter: Yes. It is like when we go and say, from a safety perspective. This is by the way a complaint that Nancy Leveson has about people that do hazard analysis. She says, *People focus on parts failing. Then how do you deal with that as a hazard?* But you can have a system where both parts are fully functional, but the way they interact with each other creates a hazardous state. One way of looking at it is you just have to go up one level and look at whoever put those two things together. It was their responsibility to understand what those interactions are and what assumptions one side made on the other. Where in some cases you may not have control over one side, like if it is a train and the platform, you do not have a control over the platform, but you have control over whether the train stops at a platform or before the platform, or before the platform before you open the door. The train does fine. The platform just fine, but the control system that understands the relationship of the train and the platform may not understand that. That is where we get these issues. That is one message that like Nancy has when she says, *There is a new class of issues that you need to discover*. We are doing a similar thing.

We also, in that context for example, one of the standards in the AADL standard suite focuses on fault modeling. In that context we have a taxonomy. In that taxonomy what we are saying is, *Have you thought about, given the subsystem fails, like a break, does it break when it is supposed to or does it not break? Or would it break at a time when it is not supposed to, so it is omission commission. It is also how hard does it break? Does it break in time?* those kinds of things. So

there are various parameters that you can go through as a checklist and say *Have you considered some of those things*? because that is what we find in many systems.

They thought about some, but in the one case it was for an engine control system. Actually getting a command to early caused this thing to misbehave. Most people think only if something comes too late then things go bad. Well, sometimes things can go bad when things happen too early.

Will: These would all seem to be important prerequisites to the notion of autonomy. Without these kinds of things being thought of, autonomy is not achievable.

Peter: That gets into a whole other ball game. That is, with autonomy they already have to have a model of the world that you are living in or that you are trying to be autonomous in. You now need to make sure that those things fit together. When you come to some boundary condition that does not fit with the understanding of that world then you need to say, *I do not trust this autonomous system, that it understands that world.* So that is pretty sophisticated stuff that still requires a lot of work. To get to some of that, at least we are taking out some of the layers below. When we build autonomous systems, the robotics folks have like a five-layer model for the whole thing.

Once you get to a certain layer where we get into the planning and understanding your environment type of thing, if you cannot rely on the lower ones, then that one does not work. *We do that, now we can focus on what we need to do up at this layer.* Some of those techniques actually go back to, for example, ideas that Luis Shaw had when he was at the SEI with Simplex.

The way he had dealt with software faults is making copies of the software that is faulty does not help. How can you get a new algorithm, control algorithm in and make sure it works? Well, what he did is he said, *Well, let me give it control. Then I watch what it does to the car or whatever it is controlling. If that gets into a state that I consider unsafe, then I am assuming this guy misbehaved. I am going to take control away from him and give it to somebody that I trust*.

This is then a way of bringing in here. It is a similar principle. Autonomy. *Let it be autonomous. If it does not work, we take control away from it or get it to fail-safe state*, which is again something that NASA has been doing forever for their stuff that they send up into the sky.

Will: So it is beyond redundancy, beyond graceful failure.

Peter: Exactly.

Will: It is anticipating an alternative path.

Peter: For those cases where you can't go into a fail-safe state that does not cause damage and then go back and get feedback from an external entity like contacting the ground or a car pulling over to the side and calling into somebody and say, *Hey, help me. Give me a driver, since I do not know how to drive anymore*.

When I watch Uber today. I mean they still have two guys sitting in there. They would not sit there and the car comes to someplace that is an area that it does not know how to do. What I would have programmed the car to do is pull over and call in and say, *Hey, I ran into something that I do not know how to do* instead of trying to drive through an intersection and cause an accident.

Will: Like in Pittsburgh when people take left turns at a time you would necessarily anticipate.

Peter: Yes. That is the Pittsburgh left for those of us that have been around long enough.

Will: And so you want systems that can learn the Pittsburgh left and can react in a safe manner. So how does this work into future lift? How is that for a pivot? Well, an interesting pivot.

Peter: Well just to kind of make it a little more of a…

Will: Graceful pivot?

Peter: A graceful pivot, yes. The AADL standard, like I said, was driven by the Army work. We always did pilot and small projects to build up the know how and how to do it and improve things. First of all, the Europeans were much more systematic about it. In 2004, the European Space Agency already had a $15 million Euro project with 30 partners that used AADL. We were doing small projects with individual citizens.

Will: They went big scale.

Peter: In 2007, the aerospace industry put together an initiative called System Architecture Virtual Integration (SAVI). It was Boeing, Airbus, Embraer and a bunch of guys. They then chose AADL as a key technology on that front. With our work with the Army we got far enough. Then we had this nice situation that Future Vertical Lift is coming up in 2020.

Right now they have this program called Joint Multi-Role. It is a technology demonstrator program. The idea is to test out some new technologies so that when they do the real product, they know whether it works or not. One part of it is on the aircraft itself: what kind of rotors to use and other things to make it faster. Then the second piece is they knew that they will not get the software on anymore.

Like in 2010, Dr. William Lewis from AMRDEC down in Huntsville had contacted the SEI. At that time he was the head of the Aviation Engineering Directorate—that is where they certify aircraft—and asked us to do a study about how to improve the qualification process, which then resulted in this four-pillar of strategy that we have about virtual integration and medal requirements up front, verifying against whatever, that whole story (https://insights.sei.cmu.edu/sei_blog/2013/06/improving-safety-critical-systems-with-a-reliability-validation-improvement-framework.html).

Given all of that, and then JMR came along, that is when they were interested in trying out this one in that context is called Architecture Virtual Integration Practice, the same concept test the other guys call SAVI. In the first round, they already had an ongoing project with two contractors trying out some technology. It is called FACE to deal with portability. We came in, in the shadow, that is Steve Vestal at Adventium. Steve Vestal is interesting because he is the guy that did the original MetaH work, which turned into AADL.

What we did is we lived off the same documents, requirements documents, design documents as the other guys, captured that in an AADL model and in the requirements specification and ran some analysis to identify potential problems, things they would run into later on. We identified a total of 85 type of things, some of them low-hanging fruit stuff, some of them relatively sophisticated things that because of the semantics we could not put our fingers on.

Since that thing then went well, the program office decided to accelerate bringing it into the program. With the next Broad Area Announcement (BAA), they then already asked people to not require them, but asked them to make use of some of this model-based architecture-centric type of work. So all six of those teams - their contracts were led this summer - are now doing work with this technology. We are in the middle of going through mentoring training and maturing technology type of work, lots of interaction right now. It has become a community effort to drive this forward.

Will: You have a very large base of people who have expertise in this already, and that is growing fairly rapidly?

Peter: We have that within the SEI. We have that outside. A number of universities are teaching it. Again in Europe, they have been much more aggressive doing some of these things earlier and here in the U.S. Part of it there is because there is this systematic funding that is going on, both at the national level and at the European level. They always require research groups working together with industry. For example, even in places like China, they have a place where they just… somebody from Europe gave a course and taught 400 people over there.

Will: We talked earlier about the early work that you did in your career and how the results of that work really are the underpinnings of what people take for granted when they learn about software engineering. It seems that this is the next wave that future college graduates will take for granted. *What we do is we model the architecture before we generate machine language that operates on the hardware.*

Peter: One of the things we had is two people that have been working with us at the SEI for quite a while. Dave Gluch, who originally was with the SEI, he then went to Embry-Riddle Aeronautics University. He then taught the course there. John McGregor is doing that at Clemson University.

In France, one of the guys that was involved in the standard is teaching at the Aeronautics University down in Toulouse. He actually runs students that are not even in the software field through this thing. There are electrical engineers, and they are using this technology and have no problem picking it up, which is kind of cool.

Some people are saying AADL is too hard or our fault modeling capabilities too hard and he just gives them the book and gives them a document and says, *OK, here is a UAV, a little one and. Go do it*. And they do it.

Will Hayes: College students flying UAVs. Neat.

Peter: This is what you find is a lot of people really want to do this stuff, know how to do it. And like in their head and those that really understand their systems, you just need to give them the tools that let them express it in an efficient way and then the analytical tools that drive it. In other domains we have had it. I mean this is why on the CAD front we have been able to design chips today.

Will: It would seem to broaden their reach instead of laboring months and months writing the base code and then struggling through to make it work, as we might have thought of it in the old days. You are really starting at a much higher, broader view of the system and specifying it.

Peter: One example of that is like—again, since we are talking about safety-critical systems, they have very good best practices in place. There is this thing called ARPs and we do a failure mode and effect analysis. Myron Hecht, a guy from the Aerospace Corporation, one thing he said is now usually there is only two people ever read an FMEA report and that is the two guys that write it. Because it is a spreadsheet type of thing that gets filled out…

Will: I have written some

Peter: Given we can do that by annotating AADL, we actually ought to generate those things. He now when he goes out and does satellite architectures, he now actually can do it as they do trade off studies. Before they only did it after all the architectural decisions were made.

This opens up new ways of looking at that whole space. This is where we get into these trade space exploration type of activities. It is on that front we actually got approved lens project for one year to look into trade space exploration, using this technology.

Will: So conquering new territories?

Peter: Yes, doing some of that. Pulling in work about trade space exploration that started out on the system engineering side. So they were doing it there. But the trade points that we have as part of the software system is a much richer set. So what we looking at is the statistical technique applicable in this context. Then on the other hand, *Can we take advantage of knowledge in the software architecture to reduce the trade space.* Because like one concrete example is, when you have a system that has multiple control systems, you tend to make sure that they all run at a very harmonized set of rates. So if you have 50 of those things, you do not have 50 points to trade. What you do is you do that in a coordinated matter. So if you can make use that knowledge. You can reduce your trade space, for example, so this is some of the stuff that we will be exploring in that LENs project.

Will: One of the things that comes with being named an SEI fellow is a grant that you can choose which of the hundreds of things that interest you and you can pursue, which 20 of the 100 are you pursuing in the near future?

Peter: Well, that is a good question, and I thought about it hard, but I actually homed in on something quite interesting. What it is, is how to deal with bias, uncertainty, and like unknowns and falsehoods in data-driven decision making?

The reason I kind of home in on that is actually quite interesting too, because when I look at safety critical systems, we make these criticality assignments, level ABCD type of thing. In terms of reliability, we pick these numbers and Guess what? When you are level A it is 10 to the minus 9 (10^{-9}). If it is B it 10 to the minus 7. Why is it 9 and 7? Whatever you have that.

In terms of assurance you get into confidence. If you talk to John Goodenough and Chuck Weinstock they have been playing around forever coming up with a confidence measure. What would be useful measures on that front. So there is that stuff there on one hand.

Then, on the other hand, I like to go to other domains and see how do they deal with that problem space. What is interesting is today we have big data. So some of that stuff is resurfacing.

There are some interesting publications that recently came out. One of them is by this woman called Cathie O'Neil, that she has a book called *Weapons of Math Destruction*.

What she means with that is people are using data blindly taking those measures. Automated algorithms make decisions that actually mess up your system rather than doing good with a system. There is another guy called Daniel Levitin, who also just came out with a new book, *A Field Guide to Lies-Critical Thinking in the Information Age*.

Will: You have a very interesting reading list.

Peter: He actually has also some interesting points. I first actually ran across him because he wrote this book called *This Is Your Brain on Music*. He is a neuroscientist. But, in addition to that, there is older work. In operations research, there is a guy called Saaty actually over here at the business school at Pitt who came up with this analytic hierarchical process.

One of the elements that he has is because you are making value-based decisions. One of the nuggets that you have in his work is when people try to determine which thing is more valuable than others as comparisons…

Will: Yes, the analytical hierarchy process.

Peter: People are not always consistent.

Will: He quantifies the level of inconsistency.

Peter: He quantifies. That's what he does. He throws it in the matrix and calculates the eigenvector. That then tells you the consistency in that data. So what I am interested in, *How do they in different domains deal with the quality of the data, the degree of bias that you have*. For example there has been some work in the area of behavioral economics where they actually have been able to quantify the bias that you have in the decision making if you only have partial access to the data sets that are available to you. Like, for example, if I sell newspapers. If I do not sell out, I know how many more I could have sold. If I sell out, I do not.

Will: That is a classic example.

Peter: Actually my son's thesis was on coming up with a statistical quantification of how much you are misestimating. And if you can do that now you can compensate for that.

Will: So somewhere in this Fibonacci sequence factors in. Have you found an application for that?

Peter: Not quite. My thought is to say, *OK, there are a number of areas like in control systems*. They know how to get noise out of data, that kind of thing. Look for the nuggets from different

domains and see if he can pull those together and then applied, and then go back to my domain and say *Where can I apply those nuggets so that when we talk about confidence measures and those other things, can we determine when there is bias in there,* when there is unknowns, and how to compensate for the unknowns and those kinds of things.

Will: A true renaissance approach to solving a hard problem.

Peter: Well, exactly. Just step out of the box and do something different. I like doing stuff like that.

Will Hayes: Wonderful story.

Peter: Again, that also is something that happened before to me when I was getting into the standards work I was working with people on campus. One of the people I have worked with is Bruce Krogh, who is actually a hybrid specialist, like a well-known person in the hybrid control systems domain. That is why I can speak to that stuff nowadays too.

Will: That does seem to be a direction CMU is going more and more obviously, integrating what has previously been thought to be disparate domains. It is not just the bridge to the computer science building there. There is the IDEATE program. I have children thinking of going to Carnegie Mellon. So this is fresh on my mind.

Peter: That is how my one son ended up in behavioral economics. He started up with [George] Lowenstein over in decision sciences. After the first year he got involved in the research on how emotions affect your decision making.

For example, when you are sad. you value things differently than when you are happy. So if you see even just a 30-second clip of a sad movie scene, you will actually make economic decisions differently than you otherwise would.

Will: So whereas we started our conversation on AADL...

Peter: Now we are off emotions.

Will: The uncertainty and being able to neatly bound things, we are now talking about very boundless areas.

Will Hayes: Well, not necessarily boundless, because if I now… I just finished teaching the course on AADL. Those guys are flying home. I gave them some examples about, for example the auto brake system and the airplane not working. They are now afraid to get onto the airplane. Clearly there are emotions involved. I am now responsible of signing off a piece of paper that says this car shall be certified to be safe.

It is not just numbers anymore. You have to take emotions into account. This is why I think this whole field of say decision science or behavioral economics is so fascinating, even to me because of that interplay.

Will: It is easy to see why you are an SEI Fellow. If you could imagine five years from now where this trajectory of work will take us, if we pick that arbitrary point in the future, what would you forecast?

Peter: It is actually something we had touched on earlier, this whole notion—at least in the context of safety critical systems that also have security requirements—is the whole notion of robustness and resilience. That is critical and you cannot do it with just pure firewalling. You need to do isolation to scope it down, and you need to have some external reference models, continuous monitoring that then makes sure that you expected system behavior meets up to what it actually does, whether that is now because of intrusion or that is because of some mistakes in software, it does not make a difference. A lot of the pieces are in place, but I think there can be some additional things that can be done that also require interaction on the hardware side.

One of the reasons I am saying that, it goes back to my history part. When I was at Carnegie Mellon we were doing capability-based systems on all the two multi-processor systems actually had hardware support for access control and everything. Intel then did a processor called the 432 that had hardware support for it. That one did not work so well, which caused them to do this project with Siemens that I was out there for. Some of those ideas showed up in the i960. It was just 30 years too early. The technology that we need today is what we had at that time already is, for example, one example of where we can make some progress in this area that we struggle with a little bit.

Will: So a richer set, more complete set of reference models too…

Peter: And in that context also continuous enforcement at all levels, not just at the outer bound. That is what gets to the robustness part and on the resilience part is the ability to understand what you expect in the world, so you can adapt to it. If you do not understand it anymore have a fallback position, like we had talked about earlier.

Will: Well, I hardly know how to stop this great conversation. Thank you very much for joining us today. This has been very interesting.

Peter: Anytime.

Will Hayes: This podcast is available on the SEI website at sei.cmu.edu/podcasts and on Carnegie Mellon University's iTunes U site and the SEI's YouTube channel. As always, if you have any questions, please do not hesitate to email us at info@sei.cmu.edu. Thank you.