System Architecture Virtual Integration: ROI on Early Discovery of Defects

featuring Peter Feiler as interviewed by Suzanne Miller

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 and operated by Carnegie Mellon University. A copy of today's podcast will be available on the SEI website at www.sei.cmu.edu/podcasts.

My name is Suzanne Miller. I am a principal researcher here at the SEI. Today, I am delighted to introduce you, once again, to my friend and colleague, Peter Feiler, who is going to talk to us today about virtual integration at the system level and some of the cost savings that have been achieved through some simulation tools, and a language that he is responsible for, the Architecture Analysis and Design Language [AADL].

I want to ask Peter to sort of give people, who have not seen you before because we have interviewed here before, a little bit of background on what it is you do at the SEI, and what is the importance of this work towards understanding how we cannot just reduce costs but reduce failures that can be sometimes catastrophic in very complex systems.

Peter Feiler: I have been at the SEI for a long time, since 1985, so it makes it 33 years now. For the last 15 some years I have been working on architectural languages that are specifically designed for embedded software systems. So they have enough semantics—especially in terms of timing, execution, and so on and fault handling type of things—so that we can get our fingers on the real issues in software system architectures, embedded software systems architectures. The way I got into that is we took some research from the '90s and turned into an SAE International Standard.

Suzanne: SAE is the Society for Automotive Engineering, which may not seem like the place that you would be an avionics (essentially) standard.

Peter: Yes, well when they started in 1906, automotive meant anything that moves.

And by the way, they officially now are just called SAE International to be exact to that point. They are actually the largest provider of avionic standards as well. If you put a seatbelt on in an aircraft that meets the SAE standard for seatbelts type of thing. Otherwise, most people know it from motor oil, for example.

We developed this language, and we have been using it to deal with issues in embedded software systems because one thing that we have realized is that the cost of embedded software systems has skyrocketed. All this cost growth is happening on the software development side, not on the system development side. In 2010 we have reached the fact that 70 percent of the total system-development cost now goes into software development. In addition, on the software development side, 70 percent of that cost is post-unit-test rework. What that points out is that it is not the individual functions that we write in software that is the problem, but the way those software functions interact.

How we make the computer resource and all that. One reason is, obviously, because when you do embedded software systems, safety-critical systems, they are very time sensitive. What we find in studies is that 70 percent of these problems are introduced during requirements and architectural design, and 80 percent of them are discovered post-unit-test. That is why we then have these percentage numbers that I mentioned to you.

Suzanne: We will do the bottom line upfront. You completed a recent study on actual cost savings by doing this kind of virtual integration using AADL. What were the initial results from that study?

Peter: Again, a little more context. It was done as part of an aerospace industry initiative called System Architecture Virtual Integration or SAVI. It was done under the Aerospace Vehicle Systems Institute, which is part of Texas A&M. Companies like Airbus, Boeing, and Embraer, suppliers, FAA [Federal Aviation Administration], NASA, SEI, and Army all worked together to deal with that problem of what I just mentioned. On the technology side, they then picked AADL as a basis for achieving that. On the other hand, since they were self-funded, they had to go back to their management to make a case for cost savings, so that those guys would be willing to invest multi-million dollars because it is an initiative that has run for 10 years. They invested over $40 million out of their own pocket into that, so they had to make a study.

The original return on investment study was done in 2009. It was actually Jörgen Hansson who, at that time, was at the SEI, and Steve Helton from Boeing, are the two main characters. I got in late in the game and the several other people on SAVI. It was that original study. It initially was a report just for the members of that initiative, and this is the public version of the report. There were some follow on studies within SAVI as well. There are articles about that, and we have references to that. The point of it was that, given the numbers that I mentioned before and the

cost of fixing errors because of delay—errors leaking from the phase they are introduced to a later stage—the cost savings are quite high when you detect them at the time they are introduced. Because if you discover them at system integration test, the cost of fixing a problem is 300 to 1,000 times higher than doing it upfront. So if upfront, you spent $10,000 fixing it, it's between $3 and $10 million on the backend that you are saving by the way.

Suzanne: Some of that has to do not with just the software itself but again, these complex interactions when you find it there, just figuring out which design elements, which requirements are the ones that need to be fixed and then all the cascading effects of doing that fix. There are so many cascading effects the later you get into system integration.

Peter: Exactly. The problem that we have is that a lot of these issues are very difficult to detect by testing. Some of them are behavioral-state type of thing. You get a state space explosion. So this is where model checking comes into the picture but then other ones are time sensitive. So for example, if you just change your scheduling algorithm, that affects the stability of your control system. If you just add another piece of software to a particular processor, which causes now…

Suzanne: A timing shift.

Peter: A timing shift. As a result of that, things are off. If you take some software that, for example, reads a sensor that before ran on a specialized little device, now becomes part of your overall software package and a partition. This partition doesn't always run at the beginning of a time frame. So you are not sampling every 10 milliseconds but off by a couple of milliseconds. You have now virtualized time as well.

So that is the other piece of it—we are virtualizing time. We are virtualizing processors and all those through partitioning concepts and various other things that we had on one hand introduced to make systems more safe, at the same time, they come with their own risks. This is part of the study that we did and then have mechanisms in the language to go after those kinds of architectural issues that really then are very difficult to trace down.

Suzanne: This really has the potential to revolutionize how these complex systems are built in the future.

Peter: That is correct, yes.

Suzanne: This idea of model-based engineering, which AADL explicitly supports, allows us to find many of these things, as you say, [that are] difficult to find in tests. Certainly if we do find them in tests, it is late in the time frame, which you get all of the effects of late rework.

In the most recent study, you do have some data now that actually supports these kinds of arguments that not only are there significant safety-, security-, quality-attribute kinds of savings in terms of defects that could end up in the field but also cost of development can be reduced.

Peter: Exactly. So when you look at the cost of development reduction for aircraft—like around 2010 on the commercial side, they were estimating like a limit of what they can afford, a limit of affordability is $10 billion on an aircraft with that cost they could save around one quarter of it on the assumption that we catch only half of the problems that are leaking. We made conservative assumptions about that. In actuality, they are actually catching more than that, but based on the conservative assumptions they were able to save 25 percent of the total system development cost, not the software development cost. But if you don't manage that, the software development cost by 2024 will make up 88 percent of the total system development cost.

We came up with those numbers. The study draws on a whole bunch of background data on these ratios, on the cost factors. There is also the question of the ratio between software development and system development cost. When we did the original study, we said the two-thirds, one-third. At that point it was based on data that each of the member companies had internally brought to the table. Then in a follow-on project, the SAVI folks, on the cost-estimation side, in the original case, we used COCOMO II…

Suzanne: Which is the common estimating method for complex systems.

Peter: …to translate the whole thing into dollar figures. Later on, they used SEER, which is another cost estimation tool. But the difference is that it accommodates not just for software but also for system engineering. It is with that one that they validated that the ratio we were using in the original study is a fair ratio.

In that context we were showing that the ratio changes to 88 percent for 2024 if we don't do anything about it. That is what led them to do this 10-year project. Meanwhile, the DoD has been embracing that too at the moment; especially the Army is very aggressive on that front. They have a tech demo program called Joint Multi Role. That is in preparation of the Future Vertical Lift program. They are now in phase three and have been using the technology. In the first phase, we did a shadow. [In the] second phase, a developer team used the technology and they are now pulling it together in the new one as well. There is real data now confirming some of the estimates that we have.

Suzanne: Excellent. You are kind of at a cusp. I talk about AADL as the overnight success story that took 25 years.

Peter: Exactly.

Suzanne: Because there was so much that went into understanding what could be modeled, what could be embedded in the semantics of the language, what had to stay outside, et cetera, et cetera, what kind of environment and tools. I know you went through a period where sort of figuring out the tooling environment and the tool chain that would allow this to be most effective was a big part of the research. You are at a place now where there is no longer a question of, *Why would we do this?* It's become a question of, *How do we do this?*

Peter: Yes. So we are moving from early adopters to mainstream. There is usually this big gap. The tools that we initially provided were open-source tools that were initially intended for a prototyping-type-of-thing. People now are expecting more industrial-strength tools. In that context then, one question is, *Well, we can turn this tool into an industrial strength or we can say*, L*et's get the buy in from the commercial industry*.

This is what SAVI was doing on the tail end, the last three years. They got commercial companies to sign up supporting this technology. One of the companies that has been very good at doing that is Ansys. With their tools they have SCADE for detailed modeling…

Suzanne: They are system tools for the most part.

Peter: Well, they are interesting because they are very good at system-engineering support, at detailed design, and then also physical modeling. But they were missing this middle layer of architecture for embedded software systems. They themselves were interested because of their customers to fill that gap. They have now with the last release a year ago, I think it was, where they are now including support for AADL as well.

Suzanne: That is a huge transition to get commercial support for these. That is, we know one of the transitioning mechanisms for getting that support.

Peter: A lot of questions that then come up is people say, *Well, we are already a SysML house, why do we need to AADL*? There is all those kinds of questions coming up. There is also the argument, *We are already doing model-based engineering*. Yes, model-based engineering when control engineers have done that a long time ago already.

Suzanne: This is another step. Virtual integration is another step.

Peter: It is another step. With SysML, you do it for system engineering. With the middle layer, that is the embedded software system architecture, this is where then AADL fills the gap. Really the combination of all three of them is allowing us to move to a model-based development process and certification process, which now brings up some other questions. If you look at the way you are contracting, how do you do RFPs [Requests for Proposals] and RFIs [Requests for Information] and proposals…

Suzanne: Requests-for-proposals and things like that.

Peter: Are we going to post a model of what we are asking for?

Suzanne: And who owns the model?

Peter: When they send in proposals, they ship us models. These are some things that we had exercised in the feasibility study on the SAVI. Currently in this next phase of JMR, we will be exercising as well. They will identify two system integrator systems, two system architects, and then several supplier teams. They will go through a whole process of a full development cycle to exercise, *What does it mean*? *How will they need to change the acquisition process? How do we change the way we are doing certification and qualification?* Instead of getting tons of paper, you get models on the government side. We now need to have labs setup so that we can handle those models, verify those models, in addition to verifying the design and the code and all that. We are not doing away with, for example, flight certification of the actual aircraft. What we are doing is running these other things upfront so that when we do flight certification, the aircraft isn't parked at the tarmac for four months because somebody misconfigured a compiler, and the code never started up on the aircraft.

Suzanne: Well that is one of the things that really increases that overall system development cost is the unanticipated cost of rework when you are that far downstream. In the case of airplanes, when you are doing flight testing, if there are any safety concerns you can't fly. Nobody else can use the airplane for their tests until you are done. There are a lot of, again, cascading effects that happen that we could prevent.

Peter: Exactly. Good that you bring that up because the cost figures that we mention here are pure system development time. There are cost savings that you have because you now don't delay delivery, so you can put your aircrafts into a mission much earlier. There are also less safety risks and the consequential cost of those things as well. Those kinds of costs are not included in some of the figures we are quoting here. So it is quite interesting.

Suzanne: So 26 percent is, we could say, is probably the lowest cost savings over the lifecycle of a product. We haven't even talked about sustainment. Once we get into sustainment, if I have a robust model that I can use to test ideas for either introducing new enhancements or correcting faults that are found, I have a whole different way of working through my sustainment issues and prioritizing what can be done. Right now things that are too hard have to wait for a modernization activity. We now have the opportunity to look at some of those things a different way. So there is a huge explosion of possibilities that this enables.

Peter: Exactly. One of the things is that we have a JMR running quite well. This is a new program where we are bringing the technology in. One of the questions is for existing programs, *Can you bring this technology to the table…*

Suzanne: For legacy systems.

Peter: For legacy systems. The answer is *yes* because one of the things you can do is use your modeling capability as a diagnostic tool. You are looking at the existing system. Many people that use these systems or maintain these systems, they create themselves a mental model of what is going on inside.

Suzanne: Sure, we all do that.

Peter: They know that certain parts are just a mess, and in certain other parts they created themselves an abstracted model, so that they get some sense of it. If we can, on one hand, capture a clean abstract model of what it is supposed to be, then say for those parts that are a mess, why are they a mess?

Suzanne: And start untangling them.

Peter: Untangling that and using the model as a structure because based on the model we now have some predictions about its behavior and explanation for why things misbehave. So you start bringing that in as a diagnostic tool. It doesn't drive the development yet, but you already are getting cost savings in the sense of the next time somebody comes with a new issue, you pull your model out and say, *Based on my model…*

Suzanne: *Here's what's likely.*

Peter: Either the idealized model—is it going to happen there too, so is it inherent in the architecture as reflected in the model—or is it just because of the part that is messy? If that is the case, we can justify cleaning up that part that is messy and an investment in fixing that. There are some interesting ways of incrementally bringing that into legacy systems as well, which we haven't systematically explored yet. But, as we have programs that are struggling and are willing to work with us with that, we can demonstrate for the legacy side as well. Then once we have data collected, we can then justify cost saving predictions with the collected data.

Suzanne: So there is sustainment coming in the future. We haven't talked about security, but security is another quality attribute that is systemic, pervasive, and has all the same issues as safety in terms of it being difficult to figure out the source of issues. Is that another area that you are looking at exploring in the future?

Peter: Yes. We mentioned timing as an issue. Safety is another one that we have explicit support in AADL to do safety. Safety tends to be primarily done during system engineering. One of the questions we asked when we started our work with AADL and SAVI is to say, *For aircraft, there are all kinds of safety processes in place. Despite the fact that we have those, why is that we have all these failures in the software*? Well, the safety process is doing a hazard analysis and your fault impact analysis and those kinds of things you do during system engineering. By the time you get to software, it is not done anymore because they are worrying about physical parts breaking.

Suzanne: Sure. Yes, which you have to.

Peter: If software has become a major source of misbehavior, we need to continue to apply that process there too. You tend to not think about it in the traditional safety sense, but you have to address it. Currently when these people do that stuff to make zero-defect assumptions for software, which can't be done.

So you have to basically assume that there will be some fault in the software. You have to, first of all, turn everything upside down. Secondly, what you have to do is focus on eliminating or demonstrating that you can handle exceptional conditions. Exceptional conditions are either intentionally to support them or simply deal with mismatched assumptions and some misbehaviors of somebody else. As we do that from the safety side, the same issue we have on the security side for embedded systems was, *They were unconnected; therefore we don't have to worry about security*. If you then put a firewall around it, that may help, but firewalls are not 100 percent either. Once you are inside, if you then can take over any part of the system, it doesn't work. You need internal protection as well.

Suzanne: So that is a different set of exception conditions that you need to be able to demonstrate that you can deal with.

Peter: Exactly. What that leads to is the same way that we are from a safety perspective. I have to protect one part of the software from another part of the software from a security perspective as well. We actually are doing a project right now—we call it Integrated Safety and Security Engineering for Mission-Critical Safety Systems—where we are saying, *There is a lot of commonality in mechanisms and analysis capabilities. We have fault-tree-analysis, and we have attack-tree analysis, what's the difference? OK? In terms of protection, we use a combination of techniques.*

Suzanne: Yes. *I have a safety exception. I have a security exception*.

Peter: Exactly. So those are some things that we are now exploring and bringing into the picture, that safety and security play together as well not just the performance-type-of things.

Suzanne: If I am a program manager in a complex system with lots of software, and I am interested in learning more about how to bring AADL and the supporting technologies and tools into my space, what are some of the resources that we can offer from the SEI viewpoint to help them with that?

Peter: Well, there is a wide range of things. We have podcasts. We have tech reports.

Suzanne: We have blog posts. You just wrote a blog post.

Peter: We have blog posts. Again, depending on the questions that you have. If it is as a manager or a program manager the cost type of thing is this ROI. There are other presentations and things that we have. We have supporting data that goes beyond the ones that we used in the SAVI notion. For example, people are bringing in data even for IT systems to confirm that the cost savings isn't quite as large because in IT systems it is largely functional issues. There are some performance issues but…

Suzanne: Different contexts.

Peter: Different contexts. We have things on that front. We are expanding on the educational front in the context of JMR. We have been developing engineering handbooks and management handbooks around this virtual integration process.

Suzanne: Right, certainly the SAE standard. SAE has a whole set of supports as well.

Peter: Well then we have. There is the standard, which is actually a suite of documents, and the standard commitment is still quite active. You just met three weeks ago here in Pittsburgh.

Suzanne: That's the second time you've met in Pittsburgh.

Peter: Yes. So the standard itself is quite active, and there is a lot of interest in the whole thing. There are courses among other things we have at the SEI. We are working on getting some of that stuff to places like DAU [Defense Acquisition University] if you are from the DoD side.

Suzanne: And the OSATE environment.

Peter: Well, and the OSATE environment is something we have always had publicly available. It's open source. It's free. You can use it.

Suzanne: For exploration purposes if you kind of dip your foot in the water.

Peter: Meanwhile, like I said, is the companies like Ansys who have commercial tools. There is Ellidiss in France who has actually two suites on that front. There is also Adventium Labs is a company in Minneapolis. One of the people who is working there is Steve Vestal who developed

in the 90s the language that became the basis for AADL. Those guys have been working with us on this JMR program and are providing all kinds of tools as well. Those are tools that are extensions to OSATE. You have to have a subscriber program where you can get capabilities that don't come with the plain level of OSATE. They, for example, have a security tool.

Suzanne: This is a completely different support environment than five years ago.

Peter: Yes, things have moved forward quite a bit.

Suzanne: If you haven't looked at AADL in five years, it is time to look again.

Peter: Exactly. The other thing that is interesting too is that a number of researchers have decided to bring more of their analytical capabilities, also code-generation capabilities. We are working on integrating the AADL capabilities with something called Future Airborne Capability Environment (FACE), which is a common platform type of thing.

Suzanne: It is another standard, yes.

Peter: Also translation between SysML and AADL to help them move between system engineering and embedded software systems engineering.

Suzanne: That has been in the works for a while.

Peter: So there is tooling on that front that is getting developed as well, so people can actually try to make use of it in the real setting.

Suzanne: I am so pleased at the progress. We have talked for many years about sort of inching up that hill. I think we are at a point where we are going to start seeing some really interesting and impactful—not just single-program results but across a wide variety of contexts. I look forward to the next time we get to talk to see what else has been very successful with this. It is a lot of work. I know it's not easy.

Peter: What is also fun is that a lot of other people that were part of the original group on the AADL are talking about virtual integration and model-based engineering types of things. I see slides that we originally developed pop up in all kinds of people's presentations.

Suzanne: Well, actually that's one of the signs of transition success. So is that our work is being referenced by others. So that's amazing.

I look forward to your work in sustainment. I am doing some work in that arena myself, and so I have some particular interest in that arena and I just want to thank you for stopping by and talking to us about this again. I guess we have to do it every five years whether we need it or not, maybe sooner next time.

Peter: I end up always having fun doing that with you.

Suzanne: Oh, thank you.

I want to thank you for joining us today. This blog post that I mentioned is available. The references that Peter and I talked about will also be available in the transcript. We make all those available to our listeners and our viewers.

This podcast is available at the SEI website at sei.cmu.edu/podcasts and it's available on the iTunes site and on the YouTube channel for the SEI. Yes, we have a YouTube channel. As always, if you have any questions please don't hesitate to send us an email at info@sei.cmu.edu. Thank you so much for viewing.