Improving Quality Using Architecture Fault Analysis with Confidence Arguments

featuring Peter Feiler as interviewed by Suzanne Miller

Time-sensitive design errors are notoriously hard to test for. In today's podcast, we will talk about a recent case study that shows how architecture fault modeling in analysis can be used to diagnose a time-sensitive design error in a control system and whether proposed changes to the system actually address the problem.

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 transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

My name is Suzanne Miller. I am a principal researcher here at the SEI. Today, I am pleased to introduce you to my friend and colleague, Peter Feiler, who has been a guest on the show before, to talk about work in Architecture Analysis and Design Language, which we call AADL, and safety-critical systems.

Before we delve into his latest research, let me first tell you a little bit about Peter. He's a 30-year veteran and principal researcher of the architecture practice initiative here at the SEI. His current research interest is in improving the quality of safety-critical software reliance systems through architecture-centric virtual system integration and incremental lifecycle assurance to reduce rework and qualification cost.

He has been the technical lead and the main author of the SAE, that's the Society for Automotive Engineering architecture analysis and design language standard, and he has received his Ph.D. in computer science from Carnegie Mellon itself. So, he's an alumni as well. Welcome, Peter.

Peter Feiler: Thanks for the introduction.

Suzanne: I'd like to begin our conversation today by having you give us some background on time-sensitive design errors. Why are they a problem? How are they different from other kinds of errors, and why are they so difficult to test?

Peter: Well, we are dealing with embedded software systems. So, that's software that controls physical entities like cars and aircraft and so on. Since they do that, they need to send control signals out in time, otherwise you brake too late and those kinds of things.

Suzanne: And, that could be very bad.

Peter: That can be a problem. The issue is, when you have embedded software, it used to be small pieces of software running on different little pieces of hardware. Nowadays, there is a set of computers in your car and multiple pieces of software running. Now, all a sudden, one piece of software running can affect the timing of the other. The problems can be shown, and actually fall into two classes. One is that an individual control signal can be missed if you step on the brake…

Suzanne: It misses that signal and it doesn't work.

Peter: The computer does not see the signal, and it doesn't apply the brake. So, those are individual control signals. But, there is also a second class that recently was encountered aboard—the FAA (Federal Aviation Administration) send out a notice—and that is for long-running things.

And one of the Boeing aircraft, they had a problem with the power systems, which would shut down after 243 days if they would not shut down and reboot the computers, and that might happen in flight. So, most of the timing are very short-notice type of things, but there is a second class that deals with…

Suzanne: An understanding where you are towards that timeline is important.

Peter: The reason they hadn't discovered that latter one earlier was everybody made the assumption that an aircraft, every so often, will get parked and totally shut down. It turns out there is always some auxiliary power running. As a result, the aircraft may actually have power on for more than 248 days. So that's what this is really all about: is trying to figure out how these kinds of errors can be found before they actually affect operations.

Peter: Exactly.

Suzanne: So, you've got a case study. That case study advocates combining the use of AADL, which is an analytical approach, with something called confidence maps to present a structured

argument that system requirements have been met and these design problems have been adequately addressed. We have talked before about AADL, but refresh us on that. Then, tell us a little bit about the confidence maps and how these two things fit together to support each other.

Peter: On the AADL side, we create a model of our system, and then virtually integrate the system, so we can test out these things. But, the testing in this case takes an analytical form where we use analysis tools or simulation before we build the physical system. In particular, in these kinds of things, we also use a technique that is very well established in the safety analysis where you are looking for exceptional conditions that can lead to hazards.

Peter: We have support for such a fault analysis in AADL. On the assurance-case side, there the focus is now on, *How do you keep track of the evidence that you produce, and is that set of evidence sufficient to convince you that we…*

Suzanne: That's the confidence map aspect.

Peter: That's the general assurance case. One particular method of assurance case is called confidence map, which was developed at the SEI by John Goodenough and Chuck Weinstock. That one is interesting because it looks at your evidence and your claims and looks for what can go wrong in it.

So, the same way we look for exceptional conditions from a safety perspective as hazards, they are looking for what they call *defeaters* in the claim, in the argument, and in the evidence. *Where can you, for example, make a mistake in the evidence, so that when you look at it, it actually is not valid?* type of thing. So, the two of them complement each other, and both of them actually use very similar techniques to get at some of the issues.

Suzanne: So, the idea is with these two techniques working together, you have got a much better chance of finding these kinds of errors than if you used either of these techniques by themselves.

Peter: Exactly.

Suzanne: So, the technical report that you published on this, and we will link to that in the transcript, you applied this approach to an aircraft engine control system. Thank you very much, as a frequent flyer, I will say. I know that a lot of your work focuses on safety-critical systems. Why did you pick this particular system, and what was significant about your findings when you applied these two techniques?

Peter: We had an opportunity to actually, for the first time, try a combination of those two techniques because the two project teams were under the same project at that time. We had a chance to work with a real customer who had a particular problem with an engine control system. They knew it was timing related. They, themselves, had used techniques, a modeling

notation called SCADE, and a tool that around it can do some verification. It's actually very good at verifying the functional behavior of this system, but it doesn't always take some of the timing-related things into account. It's very hard to find that kind of toolkit, analytical tool capability. They then knew they had this problem. They ran into it through real testing. They had proposed one design fix, and shortly thereafter came up with an idea of a second one. To them the question was *How far do we need to go to show that we really have fixed the problem or not?*

It gave us an opportunity to say, *Can we do a better job at honing in on what the root cause is, and then make the argument this is where assurance comes in*? *First of all, have you understood the root cause or are we just patching up the symptoms*. Secondly, *given the root cause, have you fully enough understood it so when the solution is proposed you can see whether the solution…*

Suzanne: And, *we can evaluate which of these solutions has a better chance of ameliorating the problem.*

Peter: So, we were able to do that on one hand from a fault-analysis perspective; use the fault ontology that comes with the error model and extender that we just recently released. And, in that context, put our finger on the fact that, *Yes, it is timing related*. It's actually early arrival of a control command can cause the stepper motor, which is a very simple control system that controls the fuel flow, to miss a step, and then show what was the assumption that was made that caused them to do that…

Suzanne: The design assumption?

Peter: The design assumption, and then come up with a way of verifying when is the design assumption not met. That then led us to evaluate designs against that and have an analytical technique of saying *Well, under these circumstances this new design will still have a problem or not*. It turns out that one of the two proposed designs was addressing one problem, but we identified a second one that they hadn't even encountered yet.

Suzanne: Oh, my.

Peter: Which happens more rarely, but still can occasionally happen anyway.

Suzanne: It's possible. Yes. Sure.

Peter: So, that was kind of an interesting exercise for us, and then to have a record of the whole thing in terms of a confidence map. It also gave us a chance to use that as an example for a proposal of some mind work…

Suzanne: The research funded work for us.

Peter: New research funded work that then came out of that initial exercise here.

Suzanne: I am very glad that you found the problem in that engine. I'm not sure if it's one of the ones that is used on the airplanes that I fly. In general, this idea of being able to model these problems, and not have to go all the way to manufacturing and test, is the big appeal of something like AADL and all of the analytical tools that it enables.

So, if I'm an organization that has safety-critical systems, what would you suggest in terms of pursuing using the kind of research that you have done here, and related kinds of research?

Peter: Around this whole thing, there is quite a large community, nowadays, that uses AADL. AADL itself is primarily a platform, so it is really the capabilities on top of it. This fault-modeling capability, fault-analysis capability, it really has shown its benefit, not just in the safety area, but we are also getting into applying the same technique in the security context.

Suzanne: You did some early work a few years ago on that, I remember.

Peter: Exactly. So, we're coming back to some of that. And, again, as we get into talking about future work, we are going to get involved in that. But, from a practical perspective, in this particular report, it shows you an example. It shows what the value was of this fault ontology, which is a categorization of certain kinds of exceptional conditions that you might want to think about. And this is how we identified one of the two that…

Suzanne: That was not already anticipated.

Peter: Exactly. It was part of our ontology, and so we had to now show that either it didn't exist, which is the argumentation part, or we needed to show, *Well, guess what? There is a circumstance in which it could occur* type of thing. So, it kind of was a cool thing to find, and being able to do that on relatively short order.

Suzanne: The idea of fault analysis is one that's been around the safety community for many, many years. I think having an otology like this that directly relates to a modeling tool is really the new piece that makes AADL even more useful, because we can couple that with things that we know are of interest to the safety community, that they are accustomed to dealing with.

Peter: That context, just a side comment is, on one hand the safety community is very good for, in particular domains, identifying what are the different kinds of faults that actually can occur in the system. What we are interested in with this ontology is to say *Given it occurs, if I ask this system interact with other systems, what are the different kinds of effects it can have on the systems*.

Suzanne: OK. So, *What are the impacts those faults have?*

Peter: What's interesting about it is that that ontology is actually domain independent and applies across all domains.

Suzanne: Really.

Peter: Because if you and I interact, it doesn't make a difference whether you are an engine and I am a battery, or you are a pilot and I'm something else. The affects that we can have is I can fail in a number of different ways. A battery can fail in 15 ways. If I'm a GPS, I can fail in 15 other ways, but the affect that I have on you is that I don't provide a service to you, and I'm supposed to, or the other way around. Or, I do it too late or too early. It is those categories that are very well established, and they are limited. Then they complement with the fault techniques of the domain.

Suzanne: That's part of what makes the analysis actually possible, is that these are not unlimited sets that you're working with.

Peter: Exactly.

Suzanne: So, it will actually narrow things down.

Peter: It nicely complements the domain knowledge that people have that come from the safety community.

Suzanne: So, where are you going next? You have got a large body of work in this area. I'm very excited about this latest piece, but I know you probably have other things in mind. So, tell us about that.

Peter: Well, on one hand we are continuing to do work in the AADL committee. We actually are going into a revision of the core standard, working on version 3.

Suzanne: Who would have thought that?

Peter: I know. I had promised myself not to do version 3, but I'm up for it now. So, we'll have a meeting in three weeks on that, some interesting things. It's mostly clean up and some improvements.

Suzanne: But that's really a testament to the value of the standard. The Society for Automotive Engineering is not the first place people think about going for software standards. Yet, it is important, AADL is important enough that it has persisted as a standard in this area.

Peter: Exactly. It shows that there is interest and value, and that people are willing to invest into making advances.

In practical terms, here at the SEI, we are doing work in several areas. One is we are actually working with an Army program to apply some of the virtual integration techniques.

In terms of research, we have one project, it is called incremental lifecycle assurance where we are building on exactly the same thing we started out here, where we are saying *How can we systematically build up the evidence*?

And one key element of the whole thing is, is evidence is only as good as the requirements that we produce the evidence against. So, you need to pay attention to the quality of the requirements as well. So, that's one of the key elements in that project.

The other piece of it is how can we do that incrementally so that we can reduce the total certification costs? We have identified three dimensions of incrementality, but that's for another talk.

Suzanne: That's for another time, another talk. Another podcast.

Peter: I know. Then we have, also, like we mentioned already, people recognize that these techniques are as valuable for security issues. Something that has actually been demonstrated in a DARPA-funded project called SMACCM under the HACMS [High-Assurance Cyber Military Systems] by Rockwell Collins and some other folks using AADL, and some of the verification techniques, to show that we can cut down on the ease with which people can break into these systems. In this one case, after they have applied this technology in a six-week period, some hackers were not able to break into a UAV [unmanned aerial vehicle].

Suzanne: Oh, that's good.

Peter: It's kind of cool. We have funding in place within our team to, again, do some additional work in that area. Where we say *Given that we have some security policies, can we now verify that the system actually is implementing and enforcing those policies correctly*? Because, in many cases, what we find is that half the time people misconfigure the systems.

So, they understand what the policy is, but in the realization data they forget to configure a certain thing. As a result, that leaves some holes for people to go in. That's the part that we are going after with this particular project. Then there are other projects within the SEI that also are combining AADL with some of the security issues.

Suzanne: That's good. Yes, because I think there's another layer of this which is analyzing the security policies themselves for the verifiability, conflict, and all that stuff.

Peter: That has been a LENS [SEI Line-Funded Exploratory New Starts projects] this year, and there's some other folks doing additional work as well.

Suzanne: Excellent. Well, you continue to be busy and that's a good thing. Peter being bored is not a good thing.

Peter: No.

Suzanne: I want to thank you for joining us today. I love getting caught up whenever we get a chance to talk about what you've been doing with AADL.

For a deeper dive into his research, and trust me there is a lot of it to look at in this area, we would welcome you to visit the SEI digital library where you can download a copy of the technical report that he [Peter Feiler] coauthored and that we just spoke about, go to resources.sei.cmu.edu. In the keyword field type the name of their report, or some of its keywords, *Improving Quality Using Architecture Fault Analysis with Confidence Arguments*. Or, you could search on the author function for Peter Feiler.

This podcast is available on the SEI website at sei.cmu.edu/podcasts and on Carnegie Mellon University's iTunes U site. As always, if you have any questions, please don't hesitate to email us at info@sei.cmu.edu. Thank you for listening.