Effective Reduction of Avoidable Complexity in Embedded Systems

featuring Julien Delange 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 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 again to, Dr. Julien Delange, who has been a guest on our audio podcast series many times. This is his first time on video, so you get to see what Julien looks like. He has talked to us about his research in the past about the Architecture Analysis Design in Language (AADL), which is very popular in Europe. It is very popular here in the avionics industry. Today, he is going to be talking about his work on a project related to that, known as Effective Reduction of Avoidable Complexity in Embedded Systems, and the acronym for that is ERACES.

We will be talking about ERACES, but let me introduce him a little further. He is researching here at the SEI. His work focuses on system and software architectures, especially in how you can model those with things like AADL. Prior to joining the SEI, he worked as a software engineer at the European Space Agency. Thanks for coming back and talking to us some more Julien.

Julien Delange: Thanks for having me, Suzanne.

Suzanne: We know that systems that run avionics, aerospace, automobiles, medicine, they are all made up of lots of different software and hardware components. They are executed on different components, and they are interconnected using various buses and protocols and different ways of getting messaging going. Let us talk about complexity in embedded systems. What is it in particular in relationship to models, architecture systems that gives us a problem with that?

Julien: First of all, in terms of complexity, we have an increase over the past years. Let's take an example that is very simple, and everybody can understand the underlying problem. Ten years ago your car has a couple of ECUs, Electric Control Units in your car. Maybe you have two or three buses to connect them. So, for example, the entertainment system, the brake control, some basic…

Suzanne: Climate control.

Julien: Exactly. Right now, your car probably has more than 50 ECUs and up to seven, eight, or even 10 buses. If you look right now, how many problems we have, not only with cars, but what you call the Internet of Things. Everything is interconnected together. In fact you have more software, more processors are interconnected, and you have a lot of complexity. Think about the TCP/IP protocol stacks. You have bugs in this stack, and you can exploit that. After that attacks the system for example, in all the Internet of Things systems connected in your home, every day we hear about…

Suzanne: Some kind of intrusion.

Julien: Exactly. A safety defect and this kind of thing because it is getting really complex. Also, we have to use some safety and good design techniques, and we do not do that.

Suzanne: It is not just the developers, it is on the user side as well. We have all heard about the example where somebody has a router, and the password is *password*. And nobody ever changes it. That provides an intrusion point that a developer does not expect to be there. It shouldn't be there, but it is there. That is a complexity, an added complexity, they were not accounting for that allows a vulnerability to occur and an intrusion eventually.

Julien: This complexity, for example the password thing, is something that is pretty simple to mitigate.

Suzanne: In comparison to other things, yes.

Julien: Exactly. For example, the password you can have the user when you install your router change the password and have a way to force them [the users] to change it. But there is something that for example, is a design, an architectural issue. For example, if your car is connected to the Internet, and the brakes are directly connected to the Internet, there is a design issue. Your braking system, should not be connected to the Internet in the first place.

Suzanne: I do not want my brakes connected to the Internet.

Julien: Exactly, but in some systems, you can have this kind of thing. Let me step back a little. We have two different kind of complexities. What I will call the *avoidable complexity* and the

essential complexity. The essential complexity is *we have more and more function on your systems, so we have to design this different function. We have to have this function…*

Suzanne: We have to account for them.

Julien: The avoidable complexity, the way you implement the function, the way you design your system, will incur a complexity. We want to reduce and avoid this complexity. We want to mitigate it. We do not want to have it because in the long run, this system has to be maintained for 10, 20, or 30 years. We do not want to maintain it.

Suzanne: There is so much complexity that it is essential. We do not want to add that extra layer of the avoidable if we don't have to. That is what we are trying to get at.

Julien: In fact, in this project, what we did is to have a measure of complexity to see exactly how you can measure complexity and what is the impact in the long term. For example, we see the impact on the development effort but in terms of money, because when you develop the system, it is going to cost more. As I said, you have to maintain for 10, 20, 30 years. We find out that, in fact, in the long run having a system that is really complex can really increase the cost.

Suzanne: It makes it very fragile.

Julien: Exactly.

Suzanne: So, for sustainment purposes, we lose sight of what decisions we made. We lose sight of *What somebody over here did to this bus*, and *I have never worked on that bus*. So all of these kinds of things make the sustainment problem even more problematic when you have complexity that is essential and avoidable.

Julien: Let me give you just an example about the impact of complexity, very simple. Take a plane, a car, you have to test the system. You have to certify the system. The more complex your system is, the more difficult it is to test it, to certify it. In the long run when you have to re-certify the system, it is going to cost you more. You know better than anybody else that certification is really costly.

Suzanne: Very costly, very time consuming, and just small changes. That is the thing. With complexity, you cannot always see the effect. You see a direct effect, but with complexity we are trying to look at the indirect effects as well. Those indirect effects are what we call *cascading effects* many times are things that we have to account for in testing and recertification.

Julien: Exactly. The idea was *Let's step back, have a look at the design artifacts, the models*—because right now safety-critical systems are designed using models—and try to find the complexity when you design the system. What we did is we made two tools. One that deals with

complexity with functional systems with SCADE. SCADE is the modeling language used to design many safety-critical systems. For example, the Airbus A380 plane and the…

Suzanne: Airbus's 380.

Julien: Exactly, the ones that go from JFK to Paris, and part of the A350 is designed with SCADE. Also using it for some military applications, industrial applications things like this. We made a tool to be able to qualify and quantify the complexity in SCADE models.

But also this is a really emerging topic because right now this industry is trying to adopt modeling tools. So we just have to have a look at what complexity means for a model. We did that, and also we designed a plug-in for AADL to try to find some complexity measure in architectural model. Again, this is really an emerging topic because people just have to adopt architectural models.

Suzanne: You have got two things that are happening together. You have got people introducing complexity just because they do not know how to model yet. Then you also have complexity that is introduced because it is needed by the design to effect the functions that are intended.

Julien: Exactly. Something really interesting is, right now the industry is moving towards modeling-based design.

Suzanne: They need the abstractions. It is too complex to try to keep all of the stuff at the code level.

Julien: We find out that training is really important when you make the move. We made a study to see how people perform and how good people are at modeling techniques. We find out that people with good experience in software engineering—with C, Java, C ++, whatever—they have a really hard time to use model-based techniques because it is really new. We try to have a sense about how difficult it is. In fact, training is really important because people have a really hard time to understand the logic of modeling.

Suzanne: Actually, you are not aware of this study, but several years ago, some colleagues and I did a study on modeling, the adoption of different kinds of modeling techniques in the software engineering community. The basic finding of it was that software engineers tend not to use modeling unless they feel that they cannot hold it in their heads themselves. By the time you get to that point, you are at a pretty high level of complexity. Now, I do have a challenge in making them able to actually use the models in a productive way. I completely resonate with what you are talking about.

Julien: This will be the challenge in the next year, being able to get engineers trained and to be able to understand exactly the logic, and how to use the tool. It is going to take a long time.

Suzanne: It is one of the transition challenges. That is one of the things that you and I and Peter Feiler, who has been leading this for a long time, have all talked about, the transition challenges on this. The tools that you are describing, where are they available to people? How can they access them if they want to try this out and see how difficult it is for them to use these?

Julien: First of all, the goal of an FFRDC [federally funded research and development center], what we want to do is to help all the people that are doing safety critical systems and provide the tool for free. What we did is we released the tool online. The Software Engineering Institute has a GitHub account, so we released the source code on this GitHub.

Suzanne: The SEI GitHub repository is the place to go for that. We will make that link available in our transcript, so people can go ahead and get to it.

Julien: The tool for the AADL ready to plug-in available, the one for SCADE, is available with documentation. Also we have a video online on the SEI YouTube channel that explains how to install the tool, how to use the tool.

The last thing that is kind of important to see how the industry is really interested in the topics is that ANSYS, the tool vendor of SCADE, took our tools and evaluated the metrics. They have an idea about using the metrics for their own tool.

Suzanne: They want to evaluate the complexity of their own tool as part of…

Julien: …to have some measure of complexity inside the tool. Like having a dashboard about complexity of the model and stuff like this.

Suzanne: So in real time, you can get real clues as to whether or not your model is increasing out of the range that you want it to.

Julien: This is goal at the SEI to transition the work. We did some research work, and right now we are trying to transition this work to industry, so it was a very successful project from that perspective.

Suzanne: Now is the next step to actually build some of the training needed, or are you looking for industry to build the training?

Julien: Right now also this work has been presented at the SCADE User Conference last month. There was a lot of interest, and there was in fact a roundtable about model complexity, and how to measure complexity. We presented our tools, and I think right now it is also the turn of the industry to pick the work.

Suzanne: And evolve it. Because the thing about measuring anything like complexity—some of our listeners will be familiar with a thing like McCabe's complexity, as a tool that is been around for decades. That measures a certain type of complexity. But it only does one type of complexity. And when we are talking about communications networks of interconnected things, there are lots of different aspects to complexity. The tooling and the measures are likely to evolve as people understand better what kinds of things actually affect complexity and which ones are avoidable.

Julien: Exactly.

Suzanne: Very good. What will you be doing next? You have got this tool out there and so now what is next on the avoidable complexity kind of horizon?

Julien: Right now, on the avoidable complexity we try to transition the tool and have some discussion with this industry to see how they are using the tool. How it can be transitioned or not? I think the really big question right now is if we need new metrics—for example, we try to apply McCabe as well as all the new metrics to all these modeling environment edge design—new metrics that are really specific to the modeling language.

What we want is to have some feedback and also have some measure about what threshold is important or not. For example, what complexity value means that your model is good, bad, what aspects you have to change?

Suzanne: Some of that has to be context-dependent because if I have something that is a relatively closed system I can probably tolerate more internal complexity. But if I have something that has many, many external connections, the simpler the individual components are, the better it is going to be in terms of…it is not an absolute number that you are going to be able to apply and say, *This is good* and *This is bad*.

Julien: It is really also context dependent. What we want is to have some people in the industry take the tool and give us some feedback about its value and what metrics make sense on that. This is really what we wanted. Also, right now, we are working on a topic that is kind of related, which is how to have some measure of security from architectural models.

Suzanne: Security is important when we start doing this Internet of Things. It increases complexity, but some of that is not avoidable because if we allow intrusion, then all kinds of things go wrong.

Julien: This is what we are working on this product right now. It is a two-year project. It started last month, so we have some time.

Suzanne: I will have to look in on you in a year or so, and see how you are doing with that.

Julien: Come back in three months.

Suzanne: You are moving quickly. This is good.

Julien: We will have some measures about security in architectural models, for example, to automatically generate a surface attack; some security tests from the architecture. What we call the security impact analysis is for each component you can see the impact if you have a security issue in the component.

Suzanne: And what it impact besides that component. The model allows you to see the effects that get propagated.

Julien: Let me give a really simple example, really basic and everybody can understand this example. Your phone can control for example your house, lights and doors and locks and stuff like this. Imagine that your phone suddenly is compromised. Then maybe somebody can take over your phone and then control your house. You see the security in your phone can affect your house, so the propagation of security issues on the system.

Suzanne: These are things that models help us to see more easily than if you are going through pages and pages of code. That is one of the reasons we do the models. You are continuing to do good things for us in this arena. I hope all of you heard the call to action, to please go out, find these tools on our GitHub. Provide some feedback if you get a chance to use them, especially on whether you see the measures of complexity that are being provided as being useful to you in your own analysis and design. We hope that you will be able to do that. I want to thank you for joining us today.

There are several blog posts for those of you that are not already familiar with AADL, the Architecture Analysis and Design Language. There are blog posts on the two tools that we have just been talking about.

So insights.sei.cmu.edu is where you would go to find links to the tools. We will be providing some in the transcript of this podcast as well, which is available at sei.cmu.edu/podcasts.

Thank you, Julien, and thank you all for listening.

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.