ReqSpec Notation

featuring Peter Feiler as Interviewed by Suzanne Miller

Suzanne Miller: Welcome to the SEI Podcast Series, a production of Carnegie Mellon University's Software Engineering Institute. The SEI is a federally funded research development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University. A copy of today's podcast can be found at the SEI website at www.sei.cmu.edu/podcasts

My name is Suzanne Miller. I am a principal researcher at the SEI. Today, I am very happy to welcome once again to our stage Peter Feiler, one of my friends and colleagues from a long way back, who is here to tell us about some of the latest work that he and his colleagues are doing with the AADL [Architecture Analysis & Design Language] and some of the derivative kinds of research that has come out of that.

Peter, welcome, thank you very much for joining us today. I know you have travel plans in the near term that will probably be taking you away from us for a while. So thank you very much for making the time.

Peter Feiler: Thanks for the introduction. Welcome to you too.

Suzanne: Tell us a little, for people that do not know about AADL, just give us a little, sort of 30 seconds what is AADL and what is important about it. Then, what are some of the challenges that you have dealt with in your work with AADL?

Peter: The focus of AADL is on safety-critical software systems like, the ones they find in cars and so on. We see plenty of evidence of issues with those.

Suzanne: I just realized, I just said *AADL*, and we have not told our readers what AADL stands for. So, why don't you go ahead and do that?

Peter: We will get to that in a moment. The problem space we are dealing with is embedded software systems, especially safety-critical. What we are encountering is that things go wrong, and we do not detect that until these systems are in operation.

There was research in the ’90s. We then turned that into an industry standard on modeling architectures and doing it through a language that has strong semantics so you can analytically assess how the system will work. We used a term *virtual system integration* as a concept to draft that idea forward.

The standard has been originally published in 2004. We have done work on it. Industry groups have embraced it, like the aerospace industry and others as well.

We also have been extending the standard suite. There are a number of annexes to the standard, and we also have been publishing revisions of the standard. The standard activities are quite alive, and its application is quite alive as well.

Suzanne: What are some of the challenges? Once you have a language like this, I am sure that as soon as you have one thing, now you need another. What are some of the things recently that have led to this most recent work?

Peter: When you deal with an embedded system, the initial focus of AADL was to characterize your system—both the software, the hardware, and the physical system it is dealing with—in such a way that you can focus on understanding the interactions between the parts. Because that is where 80 percent of the problems that we discover today do not occur until post unit test. It clearly indicates it is those kinds of things. So that has been the focus.

Five years ago, we were asked to do a study to help see how we can improve qualification of systems. That is when we realized, *doing the development is fine, and improving it, but when we are doing the qualification, we do it against requirements. We need improve the way we specify requirements, and we want to keep a record of all the evidence we are collecting, not just in the final test but throughout the whole development process*. That led to a project that deals with incremental assurance, a project called ALISA. It is in that context where we developed this language called ReqSpec or requirement specification language, which is today’s topic.

Suzanne: In traditional developments, requirement specifications—we are modeling all kinds of things—we still tend to write those in English or a native language in prose using typical *The system shall do this, that, or the other thing*. What kind of information is lost when you use that kind of requirements basis instead of doing something that is more model-based, and it has some stronger semantics?

Peter: Again, studies have been done on that front. What we find is that a lot of the issues are introduced during the requirements phase, and it is often because they are missing requirements. There is ambiguous requirements. There is just…

Suzanne: There is a lot of assumptions. I know when I wrote requirements, I am creating a mental model of what I want, and I have English prose as my way of expressing that. It is very difficult to express essentially what we would put into a model when we are using that kind of a prose expression.

Peter: Exactly. There are a couple of ways forward. One way where there has been quite a bit of research is to process textual notation and try to extract from it the behavioral part as a state machine. Then, once you have a state machine, you can figure out: *Are you missing some states? Are you missing some transitions?* But, that is just one piece of the puzzle.

What we have said is, then, *complement that research and leverage* obviously but to say, *Can we come up with a technique that helps us make sure we have coverage*? Because like I said, *80 percent of the problems are not necessarily on the formal part but simply you missing certain things*. This is where we then have an approach, we call it architecture-led requirements specification, and I will be elaborating on that when we get to the use scenarios.

Suzanne: So, the SEI—I am just laughing a little bit because we have many things now that are architecture-led, architecture-influence. Many of us at the SEI strongly believe that architecture is a key communication vehicle. It is just like, every time we do something else we run back into that, is actually one of the core elements that we have to deal with. So tell us a little bit about the requirements specification, ReqSpec notation. How is different from textual processors for state machines and other things that people have tried?

Peter: We got into this notation by building—again, I like to build again on other people's work so I do not have to reinvent the wheel. It turned out there was a group, a research group in France who had been looking over that requirement specification space because again, there was research in the late '90s and early 2000s on that front. They were building on that, and it looked also at what people do in SysML and came up with a proposal for a metamodel for requirements specification associated with AADL models and brought that to the committee.

In that metamodel, what they wanted to do is focus on the ability to specify stakeholder requirements, or sometimes [they are] also called goals, because it is goal-oriented that requirements engineering came from, and system requirements. The main difference is—and you want to have an explicit separation of those two—stakeholder requirements, they sometimes can be in conflict with each other. And you do not necessarily resolve them and those kinds of things. So you need to give space to express everything, and you want to make sure that everybody talks about all the requirements and assumptions.

Then you turn them into system requirements, and system requirements need to be verifiable, because otherwise you cannot qualify against them. So, in our notation we support both of them.

For both of them, we do it by associating them with a specification of your system. In some cases you use the architectural language not to describe the internal architecture of the system but the system in its operational context, because that is a system as well.

Suzanne: So, you are including the environment then as part of what you are describing in the *requirements envelope* I will call it?

Peter: Exactly. When you do that…I have seen plenty of these documents for—on the stakeholder side, the first describe use scenario. That is, for example, for a situational awareness system. *I need to track electric power lines. I need to track other flying objects. I need to track mountains*, and this and that and the other thing. Then, at a later stage, when they elaborate those requirements, they forget half of them. I mean it is simple things like that. They find out a model where I say, *Here is my system and here are all the entities in my environment. And I will have something specific to point to and make sure I have covered each of those and have covered them from a variety of angles.*

Suzanne: So you are basically doing a model of the environment in which the requirements are meant to satisfy goals as well as modeling the goals and then as well as modeling the expression of how you think when you start the system is going to be able to actually achieve those goals.

Peter: Exactly. There is a number of scenarios that we make use of if you already have a stakeholder document that was collected separately, and it is a paper document. We can import that into our representation and then you use that as a basis to start focusing on how to define a system requirements. You do those around now your architectural specification.

If you don't have stakeholder requirements yet, you may already at that point want to define those in the context of an architectural representation.

Basically we have two formats for the requirements: One that follows a document structure and then another one that follows the architectural elements. You can interlink those so there is full traceability all the way back to individual stakeholders.

Suzanne: That also enables a different kind of discussion about what you can get into. I do a lot of work in Agile, and the Agile requirements way of dealing with things often is called *a story* where you add into the *what*, a *who*, and a *why*. This is a different way of getting at that *who* and the *why*. You are dealing with the stakeholders explicitly. The *why* is coming from the context. If I can't figure out the *why* directly, then I am probably going to ask about it. Now we can have a discussion about what we really drives this, find a new element that we did not know about in the architecture, and all those kinds of things. Excellent.

Peter: Then the second part of it is, you have the requirement for the system as a whole to evolve your architectural design. Are those requirements moving along to the various subsystems or not? Most of the time, they tend not to.

Suzanne: Something happens.

Peter: By using our notation, it is a very natural…It is the same mechanism that lets you say *OK, I am going to do my next-level design. I now define the assumptions that I make about the parts I am using*. Guess what those assumptions are? They are the requirements I have on the parts to be used, or they reflect the spec sheet of a component I may have chosen.

That is how, then, this whole process becomes an incremental process of pushing requirements down and at the same time, as soon as you have done the next level of design, based on those specs of the next level, I can already verify whether the upper-level requirements are satisfied at this level, and so you get this continuous incremental assurance activity.

Suzanne: So this has been part of your real focus, is getting things in place that allow not just virtual system integration but the incremental assurance, which is a very expensive part of certainly DoD systems and other complex systems. This is a way of enabling, going up a little bit higher in that space, by having something that is verifiable all the way up to the stakeholder requirements. Nice.

Peter: Exactly. It is fun work to do. As we have been developing that, we had the opportunity of actually applying it with an external customer. There is an Army technology demonstrator program called Joint Multi Role. They had a project where they were leading some work with two contractors and some internal integration team. We then came in part way into the project. We then did a shadow project, took the requirements that were given to those contractors, captured those in this notation. Then in that context, again early on, tried to see if we can identify some holes and some potential integration issues. That was then validated. We identified about 85 issues in the requirements specification. It was confirmed by the contractors, *yes those are the issues that we have to resolve.*

As a result of that, now in the next round— they just had a request for proposals and are currently leading the contracts. They will lead six contracts. They did not require the contractors to use it but they strongly encourage contractors to use this whole concept of virtual system integration. In that context, we are offering as part of the tool set that comes with AADL the OSATE tool set, the ReqSpec support is already included.

Suzanne: I was going to ask you about that.

Peter: It is already publicly available for people to use.

Suzanne: If I am a software architect or if I am a requirements business analyst, and I have already been using the tool set in OSATE, I have already been using AADL, I can extend easily into this.

Peter: Exactly, and it already comes with the standard release now.

Suzanne: What other kinds of support do you have for people that want to learn about this and want to become proficient at using ReqSpec?

Peter: One of the things that we have is a tech report that just came out that elaborates on all the elements of the language, because there is a rich set of capabilities underneath it. To just give you another example of how we try to home in on semantics of these things and sometimes the solutions. The guides that came with metamodel distinguished between *verifiable requirements* and *satisfiable requirements*.

Suzanne: How did they distinguish those? That is an interesting set of words.

Peter: They use those words, and I think one of them was traced back to the Sysml crowd. But *verifiable* means yes or no. There is no question.

Suzanne: It is a binary.

Peter: It is a binary decision. *Satisfiable* is *I would like to you to build a car for me that goes 250 miles an hour.* It is kind of a design goal.

Suzanne: I am sure you would, Peter, but get over it. Maybe 120.

Peter: And given that you have conflicting requirements.

Suzanne: I can do there is some room for negotiation.

Peter: It is a goal that you set and then what you do is as part of the process of taking stakeholder requirements and turn them into system requirements, you then kind of make a trade off.

Suzanne: It is a trade-space kind of requirement.

Peter: What we have now is actually support in that notation to not only specify requirements that are a must but also design goals. *This is what I want. There is a minimum, but this is what I would like to have*.

Suzanne: We sometimes call those *thresholds and objectives* in DoD language.

Peter: Exactly. People use similar terms. It is stuff like that, and you have a notation. The other piece that is around it is part of the methodology of doing this thing is to say…First of all there is all this traceability stuff all the way back from the system requirements up the architecture hierarchy back to the stakeholder requirements and forward into the implementation.

We also have some support in there that allows you to analyze the specification for coverage. So, in coverage we do it along three dimensions. One is along the different elements of your system model. When you do stakeholder requirements and you say *I have a system that deals with 15 different types of elements in my operational environment. I now need to make sure they are requirements for assumptions you are making about each of those.*

Suzanne: Because they are all going to have different effects on the system.

Peter: Exactly. If that is 80 percent of the problems, we do that. The second piece is to then say, *Can we make sure that you are talking about all different interesting or quality attributes, especially operational ones, that are important to you*?

Suzanne: And these are the non-functional requirements that pervade the system.

Peter: Yes, non-functional properties. I specifically then call them *operational* ones because those are the ones that deal with the operation of the system, like safety and liability versus *Is this thing maintainable or is it modular*?

For that, we are just incorporating elements that come from other work at the SEI, namely from ATAM for example, the utility trees. They give us a very nice framework for saying *Have you talked about security? And under security have you made the requirement concrete enough that it is measurable*? *Because we need to verify it* type of thing.

The third part of it is— and this is where the safety criticality comes in—is we make use of the Error Model Annex of AADL—and I think we have a blog on that before. Or, if not, we will be doing one shortly. As part of that, we have a taxonomy of fault propagation.

Suzanne: A typical fault propagation.

Peter: And typically what people think of—is the thing works or it does not? But there is—that is many more variations on that theme so we, again, can then make use of taxonomy to make sure and point out to you, *Hey, you have a data stream? Have you talked about data rate problems? Have you talked about value problems? Have you talked about time-sensitive data problems?* Those kinds of things you can address as well.

Suzanne: That is a domain-specific model. That is an aeronautical-systems model. In the future, there could also be medical device models. There could be other domain-specific models.

Peter: Actually, the taxonomy is for the kind of impact that it can have on a fault, and that is a very generic set. It is a limited generic set. There is a second class that is then domain specific. If I provide you with a subsystem, a black box sub system, you do not really care of how it fails internally. If I am a GPS and I am supposed to provide you with location data, what you care about is, *Do I provide it to you or not? Do I provide it to you too late or too early? How accurate is it? Do I send bad data? Do I send it at the right rate?* It is that kind of thing that you care about. And how internally, of course, that is obviously a domain-specific problem, but the taxonomy focuses on the impact because that is the interaction part that we typically struggle with. People know their domain. I am leaving them to their thing, but what we do is now look at the architectural impact of things going wrong in special cases.

Suzanne: And things going wrong in operations.

Peter: In operation. What you have is you have exceptional conditions. You want to make sure that they are handled, and we document which ones do you assume do not occur, and which ones do you assume do occur that you are willing to handle?

I use the word exceptional condition, or fault condition, because what we then do in our current focus has been on the safety implications, but with more recent research work, we said, *The same exceptional conditions have also security implications*. That framework is extensible not just into safety but applicable to security as well.

Suzanne: And that is one reason we call it incremental assurance, because assurance is not just security or just safety. It has multiple dimensions. Many of them share attributes that you can take advantage of in these modeling kinds of activities.

Peter: The other part of it is, we already said is, that *Requirements need to be verifiable*. The way we do that is then as part of the specification, that is not part of the ReqSpec language but a set of other notations that are being done under the ALISA project. We will be talking about that at a later stage—Iis you have a verification plan where you identify what are different verification activities that need to be done. With verification activities, it is not just manual things like *Look over this document*, but as running analyses on these actual models.

Suzanne: Particular analyses.

Peter: Yes. And given we have that specification in place under the ALISA project, we have then automated the execution of these things as well and the automatic generation of assurance cases as kind of an evidence type of thing as well. Like I said, it is beyond the topic of today but…

Suzanne: Sure but this all connects.

Peter: One of the reasons we want to improve the quality of the requirements is because that is the thing that we verify against. There is a clear link by *saying, If I define the requirement, I also want to define how I am going to verify it. It helps you make the requirement sharper*.

Suzanne: I think a key thing here is you are finding a way to get to missing requirements. If I verify against the requirements [that] are there, but I am not verifying against requirements that I do not know about, I am missing. Those are potential fault areas that are gaps, that are very important. This is a way of minimizing some of those gaps.

Peter: I referred to the tech report, but there are also reports about the work that we have done with that army group that I mentioned [see reports here, here, and here]. It illustrates that particular example. There is actually a total of three reports that are specific to that work. There are SEI reports, SEI special reports that are available, as well as the report that documents the language itself.

Suzanne: I am sure that many of our listeners will be very interested in those. And I am interested in them, [as] someone that used to write requirements, and I want to see what the latest is on requirement specification. Back to the ’80s when I was doing this regularly, it was so painful and always with so much pressure because you knew that nothing—that you could not make it perfect.

And we have been acting in many ways as though we could make it perfect, but we know we cannot do that with pros. We may not get perfect with ReqSpec, but we are going to get a lot closer with that, I think, than we can with prose.

Peter: Exactly. It is a language that is a textual language, so you can process it nicely and everything. There is a metamodel behind it. What is going to happen now is we are taking all the textual language back to the AADL committee, because standardizing around the metamodel is not sufficient. We needed something more. And so take it back to the committee so that this language now will become one of the annex standards, like the aero model, as well. It is one of the things that I am—one of my next steps—just specific to the ReqSpec.

Suzanne: So ReqSpec is well on its way. What else is coming? You have got the incremental assurance work. What is next in that sort of whole range of things that are related to incremental assurance?

Peter: Like I was mentioning, we have a language that lets you annotate verification plans. The whole thing is compositional, so you can do it large scale systems. We have an implementation of the whole thing. In terms of verification activities, we support a wide range of things that can be used to specify the analysis.

We have incorporated technologies from other groups like Rockwell Collins, for example. We are doing that under a project, which will close out in September. In August, we will actually have a public release of these verification capabilities as well.

Other work that we have is, we have an ongoing [project] where we took this work into the security arena. There what we focus on is to say [something?] is given, have a specification of security policies maybe. For example, that certain information is only accessible to certain people. Or, if you are in an embedded system and you are supposed to be able to switch on and off the light bulb. You are not supposed to reprogram the router built into a light bulb and the Internet of Things.

Well, today there is no clear specification and no enforcement. What we are doing in that project is to say, *If you have such a specification, if you then look at the implementation are those policies actually enforced?* [That] type of thing.

Suzanne: You can do this now in a virtual way by having a good model of the system.

Peter: Exactly. Again, it is a two-year project. We are at the end of the first year. We have already some interesting results. What we have come up with a [taxonomy] surface identification and a [taxonomy] tree generation and analysis. Also a generation of implementation against a verified kernel, so that we can produce implementations that are fully secure. We are in the middle of releasing those tools, actually, probably at the end of this month as some work.

Then the final thing is something that we are heading towards is—and I just am working on a proposal for that—is to support trade-space exploration. When you design your architecture, you have a bunch of options. How do you figure out what are good candidates if you have a large system that has so many different places where you can make choices?

There has been some interesting research at the system-engineering level where people have used some techniques, some of them a combination of statistical and others for things like satellite architectures to explore new ones for aircraft and so on. We are taking those concepts and now applying for the embedded software system as well as the proposal that we just started figuring out. It is, again, a nice extension. We just try to build on what we have and expand out into additional areas.

Suzanne: One of the interesting things that comes to my mind to me when we talk about that is there are different points in the evolution of your architecture where you know different things. You can't do a whole trade-space analysis at the beginning, because you are not going to know enough to be able to make decisions. If I have got incremental model building, incremental requirements, I can do incremental trade-space analysis as I am imagining at some point you will

be able to flag, *Hey we have this data now so this kind of trade analysis that we could not do six months ago, we can do now*. That is very exciting.

Peter: It is kind of interesting, too, because you find different classes of trades that people are interested in. In some cases, they just want to tweak their rate at which they are processing things. In other cases, they want to add some functionality or not or pick one vendor over the other. Then there are some choices that have major architectural implications where we bring a new concept in, like, *Should I go partitioned or federated? Should I use a pull protocol or a push protocol. When I produce data immediately shipped, or do I request the data before it is being shipped*. [That] type of thing. We have analytical techniques that you let you find out what the implications of those choices are.

Suzanne: Another light bulb just went off for me. A lot of the slowness in adoption of parallel processing has to do with some of those architectural choices and not being able to know until you implement what the effect is on total performance. When we have the ability to model several processors in parallel, we can make some of those analyses much more effectively.

Peter: To come a little bit to a closing, too, but also why I am still excited to work at the AADL committee. By the way, my next trip is actually to the committee in Germany next week. We actually have a strong group of researchers that come to these meetings because they have seen this as a very interesting platform to drive their research. By doing that, [they] immediately have a reach into companies that are trying to use these technologies…

Suzanne: It is a very short transition space.

Peter: That is where you have groups that get as some of these issues. So, for example, right now there is a group of folks led by Jean Pierre Talpin, from INRIA, that are taking the concept of synchronous systems and expanding it to allow [it to] stay within the formal framework that they have but still allow more flexibility to deal with more concurrency, more asynchronous system behavior, and things like that. That is what makes it exciting, is you have one foot with industry and the other foot strongly embedded with the research community.

Suzanne: Well as always, Peter, I love catching up with you and finding out what is in your head and what has actually made it out of your head into the world. We started this journey…

Peter: A long time ago.

Suzanne: I do not want to say how long ago talking about model-based engineering and the role of a language like AADL and the fruits of that are absolutely wonderful at this point. I see just more and more coming into the real part of the world. That is very exciting.

Let me tell people about where things are for them to find, because you have things all over the place. AADL itself, if our listeners go to aadl.info, you will find a wealth of resources and links to other resources.

SEI also maintains a GitHub repository with downloads of different elements of the language and other information. Of course, with this transcript, we will provide a list of resources, because there are several that Peter mentioned. We want to make sure that everybody has access to those. This was a very rich discussion. Thank you very much Peter for joining us.

Peter: You are welcome.

Suzanne: I do want to highlight the fact that, as always, you can contact us through info@sei.cmu.edu. That is another way you can get information that we may have missed in our resource list.

As always, this podcast is available on the SEI podcast at www.sei.cmu.edu/podcasts. It will be available on the Carnegie Mellon University's iTunes U site. Thank you very much for joining us today.