What’s New With Version 2 of the AADL Standard?

featuring Peter Feiler interviewed by Shane McGraw

Shane: Welcome to the SEI Podcast Series, a production the Carnegie Mellon Software Engineering Institute. The SEI is a federally funded research and development center at Carnegie Mellon University in Pittsburgh, Pennsylvania. To learn more about the SEI, please visit our website at sei.cmu.edu. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts. My name is Shane and today I'm pleased to introduce you to Peter Feiler, senior researcher the Software Engineering Institute. In today's podcast we will be discussing the latest developments with the Architecture Analysis and Design Language (AADL) standard. Before we dive into the topic, a few words about today's guest. Peter is the technical lead and author of the Architecture Analysis and Design Language standard, which was published in November 2004. Version 2 of the standard was published in January 2009. He’s a senior member of the technical staff. His research interests include dependable real-time systems, architecture languages for embedded systems, and predictable system analysis and engineering. Welcome, Peter.

Peter Feiler: Thanks, Shane.

Shane: Let’s start off by having you tell us a little bit about the Architecture Analysis and Design Language standard and its role in software-intensive systems.

Peter: Sure. First, let’s talk about, “What is a software intensive system?” Today almost everything is software intensive. If you have a car or an aircraft, they don't work anymore if the software doesn’t work. My wife just picked up our car from the repair shop, and half of it was software-related issues. What we are finding is these systems make use of software, not just in individual parts, but it’s the software of the different parts working together that makes the systems work. Well, because they have so many problems, in the aircraft industry area they recognized that problem in the late ’90s already.

In that context we developed AADL as a standard under the SAE International. SAE originally was called the Society for Automotive Engineering. Automotive meaning not just cars, but other things. They're actually the largest provider of avionic standards. So, we developed that language. The idea is to characterize your software-intensive system: the software architecture in terms of interacting tasks, how it's mapped onto the computer hardware, and how it interfaces with the physical system that it actually tries to run or drive. By doing that we can then early on do analysis of different quality attributes like safety, reliability, performance, and so on. So, that's the primary goal of the AADL.

Shane: Okay. So, what are some of the benefits from that? Does this lower costs in development or maintenance?

Peter: Well, good question. What industry has found is that—I'm going to use an example out of the aircraft industry—so, for example, 70 percent of the system cost was software-related and 70 percent of the software cost was rework costs, because they found problems late in the development process. Other studies have shown that 80 percent of safety-critical software errors are introduced through requirements and architectural design; 70 percent are introduced there, and 80 percent of those kinds of errors aren't found until system integration. By that time the rework cost is much higher than if you would find it early on. So, there is definitely major cost reduction possible, and the cost has gone up exponentially in the last years. That's why the aircraft industry now has an initiative called system architecture virtual integration (SAVI) in which they are trying to tackle this problem. When they looked for good technologies to drive this forward, they picked AADL as one of the key technologies.

Shane: Peter, you're the technical lead and author of AADL. Can you tell us a little bit about your role in developing it and the history of it?

Peter: Sure. The AADL has its roots in some research technology of the '90s under several DARPA (Defense Advanced Research Process Agency) programs, in particular two architectural languages. One, called MetaH, was developed by Steve Vestal at Honeywell. And the second one, called Acme, [was developed] by Dave Garlan here at Carnegie Mellon.

MetaH was interesting because it specifically was focusing on embedded systems, and so a lot of MetaH shows up in AADL. On the other hand, MetaH was a closed language. You couldn't make extensions to it. That's where Acme came into the picture. What I did as the technical lead for the AADL was combine the concepts of those two languages into a single language, which we call AADL. Now, the work for the standard started in early 2000. At that time we had to have several companies that are members of the SAE actually approve a committee to pursue this as a standard to show that there is a need. So, it was very much needs-driven rather than "Here's a technology. We are looking for a nail." so to speak. So, we started in 2000.

In 2002 we got two groups from Europe to join in. One is the European Space Agency, because they were looking for the next generation, moving their technology forward, for building systems. The second group was a set of French companies and research institutes centered around Airbus, because Airbus obviously is a big market. They were looking for moving forward. In both cases they identified the draft AADL standard as something that looked very enticing to them and wanted to participate to contribute making it work well. What happened then was that the European Space Agency had a funded project that was going to start at the end of 2004, which forced us to stop fiddling with the standard and put out a first version. That's why 2004 [was the first version], and then we took the five years of experience that these groups have had with the AADL to revise the standard and come out with a second version in 2009, which is the topic of this report.

Shane: Excellent. So, you mentioned the technical report, which you co-authored in March of 2012, along with Joseph Seibel and Lutz Wrage of the SEI. The official name of the technical report is, What's New in Version 2 of the Architecture Analysis and Design Language Standard. So, tell us a little bit about the research and what issues the report addresses?

Peter: It talks about what has changed since the original part of the standard. The reason we have the three authors is that I was responsible for the language. Lutz is the guy that maintains the Meta model behind it in the XML-based interchange format, which allows you to bring in tools. Joe Seibel was on it because he was the guy that did a lot of the tooling implementation work in support of the language.

There's a toolset called OSATE, which the SEI had originally funded to support the original version of the standard, and we have an upgraded version to support Version 2 of the standard. For what's in the report, it helps you understand new things we have introduced into the standard. There's new things in a couple of areas. One is several new concepts beyond what we already had in the original language. The original language had concepts for the software architecture in terms of threats, processes, and subprograms, but also hardware-related things like processor bus, memory, and system. Then things like ports and access mechanisms to talk about sharing of resources and things like that. Based on the experience, we introduced a couple of new concepts.

One is called virtual bus. It allows you to talk about protocols and protocol layers or virtual channels, kind of an abstraction of the physical hardware. We wanted to be able to expose that to the application because it depends on those things without having to give the details of the underlying platform. In a similar manner, we introduced the concept of a virtual processor to play a similar role. That's to represent the notion of partitions of virtual machines or hierarchical schedulers.

The other thing we did is we made explicit the notion of an abstract component, so we can talk about an early design without having made a decision what concretely it should be. And, we allow you to specify things in the parameterized manner, so you can do templates and patterns. So you can do partial models and then instantiate them more efficiently.

The final thing that we did do is we put in support for arrays of components, that is if you have a lot of entities of the same kind. One reason we did that is because we had people that are dealing with sensornets come to us and say, "Well, we have 100 by 100 sensors. We don't want to define 10,000 sensors one by one." It's those kinds of things. Basically improvements that were coming again, purely based on end-user needs and the application areas that they used.

Shane: So is there a user community? You mentioned end users. Is there a user community that people can reach out to, or, you know, are these clients of the SEI?

Peter: The user community of the AADL is some of them are clients of the SEI, and it's an international community. We have people in Japan, in Europe, and the U.S. There is on one hand research institutes who are interfacing their analytical capabilities to AADL because that was part of the idea, not for AADL to give you all the analytical capabilities, but to be the source of information for driving these analyses. So, it gives a chance for those people, for example, people who are doing model checking, to now associate their capabilities with a modeling language that is used in industry, and therefore they can now apply to industrial research settings. On the other hand, industry is figuring out how to use that on real-scale projects and one example of that is this whole industry initiative that I mentioned before called System Architecture Virtual Integration.

Shane: Excellent. So, what do you see as the future of the AADL language? Are there plans for a third iteration, and if so, what do you expect those to be?

Peter: From the standard perspective, the core language, the basic AADL, we don't expect to have a new version out. We just did a minor revision to the Version 2 standard to fix some errata that we had missed when we published it in 2009. The effort of the committee goes into extensions to the core standard. AADL was made extensible for areas that the core language does not specifically address. The core language addresses performance and timing-related things, but if you want to do safety for example, or security, you have mechanisms that it can add into AADL to then address those. Some of those have been standardized. So, we have a standardized extension for fault modeling called the Error Model Annex. It was published in 2006 originally, and it's currently in revision to improve that.

We also have one for describing behavior of the system, interaction behavior. We have one to deal with partitioned architectures according to a standard called ARINC 653. And then there's several in the works, one of them, for example, around requirements, definition, and analysis.

And another one has to do with constraint languages. Since you're getting to the closing, I wanted to give people a couple of pointers where they can follow up on information. There is the general website called www.aadl.info. For a more up-to-date website and also to see what other people, the community, is doing with AADL, especially Version 2, there is a wiki that has more up-to-date information that has presentations that people give at our committee meetings (where they give us feedback) and also sources to tool sets that people have made available, extensions to the core tool set that we provide. That website is wiki.sei.cmu.edu/aadl.

Shane: Excellent, Peter. Thank you very much for joining us today. This was great information. Once again, for more information on this research, I wanted to also mention your technical report again, *What's New in Version Two of the Architecture Analysis and Design Language Standard* and that can be found in the SEI library, which is sei.cmu.edu /library/reportspapers.cfm.

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