Reducing Complexity in Software & Systems

featuring Sarah Sheard 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 very pleased to introduce you to my colleague and friend, Dr. Sarah Sheard, a fellow researcher at the SEI. In today's podcast, we will talk about research that aims to reduce complexity of aircraft systems and software. Before we begin, let me tell you about our guest.

Sarah is a senior engineer in the SEI Software Solutions Division who has authored several publications on systems and software engineering in the sustainment phase as well as other phases in the development cycle and has helped the Air Force with their software engineering manual. She has been with us since she joined the SEI in late 2012. Her current work includes both research in the fields of systems and software engineering and application of SEI technologies for government customers. She has expertise in cyber-physical systems, measurement and analysis, lifecycle technical support, sustainment, system security engineering, system of systems requirements, architecture, software-development processes, and systems-engineering processes. She is a Jill of many, many trades. So, she has got lots of expertise to share with us. Welcome, Sarah.

Sarah Sheard: Thank you.

Suzanne: Let's start off by talking about the growing complexity of systems in software and why we are concerned about this.

Sarah: Well, Suzanne, everybody knows that things are getting more and more complex. It's just something that we have a gut feeling about. As we put more software into our products, the software gets complex for a couple of reasons: One, it uses yesterday's software as well as adding today's software. And, two, the software has to connect to other software.

Suzanne: And, that software may have been built yesterday or today. So we've got this…

Sarah: Or tomorrow.

Suzanne: So, we've got this combinatorial of different systems that have different legacies but all are trying to work together.

Sarah: We may be expecting our software to interface and work with software that hasn't been built yet, that hasn't been defined, would be in different languages. So, that is a much harder problem. In general, people talk about complexity, but they don't really understand what they mean when they talk about complexity. So, we were trying to figure out what that [software complexity] means. In particular, in its relationship to aircraft avionics.

Suzanne: OK. So, what about aircraft avionics makes it particularly useful as an area to deal with the subject?

Sarah: Well, there was a recent incident where an airbus plane went down. It was a military aircraft [that] went down because someone didn't load the software configuration correctly. We have known in the field, or, had gut feels in the field for a very long time, that software has ways to go wrong that, we can find when we look at them in reverse, we are seeing how they got there. But We can't predict them from just looking at the software.

There are just so many steps that have to be taken, for example, in loading software like Airbus did incorrectly. But, there are so many possibilities of what might happen, depending on the actual values of the data that the software is working with. We are concerned that, one of these days, we're going to have a problem that we didn't anticipate and that is going to hurt a lot of people.

Suzanne: That is rooted in software.

Sarah: That is rooted, well, within the software or in the systems that involve software, like the avionics system. The amazing thing in my mind today is that we haven't had more of these. I'm very impressed at how safe flying is. I'm just afraid for the future a little bit.

We want to proactively take a look into what could go wrong. What, in particular, about complexity…Can we tell how complex something is? Can we tell, given that value—whether it's a numeric number or perhaps a qualitative number—does that tell us anything about how sure we are that the software will be safe, and the systems?

Suzanne: And, how can we establish that confidence?

Sarah: Exactly.

Suzanne: OK. So, you started research into this area. What was the impetus for this particular piece of your research?

Sarah: Well, the FAA [Federal Aviation Administration] actually came to us and suggested that we might be able to solve this. We have had some work with them before that touched on this area in terms of virtual integration of software architecture. And, they asked that we would work on the complexity. It fell in my lap because of my recent doctorate on complexity and systems. So, it turns out it is quite an interesting extension from that work looking specifically about what about the software is complex.

Suzanne: So, according to your recent blog post on this research, you recently completed the first two phases of this research. That was a literature review and an initial look at metrics, measures related to this. Tell us about some of your findings.

Sarah: The interesting thing about the literature review, which was intended to find all the definitions of complexity out there and then figure out which ones related to systems and software, was that nobody had defined it, OK? We found that out. Nobody has a definition of complexity that's more complex and/or useful than say Webster's [Merriam-Webster's Dictionary]. But, when we looked into it, we found that there were a couple of aspects of complexity that kept cropping up and that were important. One was the causes of complexity and one was what I will call the effects of complexity, or the impacts.

So, we did quite a bit of work, a taxonomy actually, of what kinds of causes cause something to be complex. Whether it's software or even in systems. And, then, what impacts it has on you, the designer, the operator, the user that the fact that that thing is complex. So, we ended up realizing from the literature search that we really had to not talk about a definition so much as what is it.

Suzanne: What are the attributes?

Sarah: *What are the important attributes of it?* Correct.

In the measurement phase, clearly, the idea is to understand how complex something is. Independent of this work, prior to it—and it's being reinforced by the work I am doing here—I am realizing that you really can't have a measure of complexity. The reason is, measurement is simplification. So, if I say, *How big am I?* I could give you my weight, or I could give you my height. That would tell how big I am. But, it won't tell me until I go to the store and try things on what size I take. It won't tell me whether I'm healthy in that size or not. So, it is a simplification where we get a very simple number that represents something more complex.

Now, if you take something like complexity and try to simplify it, are you not trying to measure something in terms of its opposite? So, you can't measure red by saying how green it is. And, I don't think you can measure complexity by saying something simple about it. That said, complexity is a notion that we all have, and something about the complexity we can measure. So, we can measure say, algorithmic complexity, or we can measure McCabe complexity, or we can measure any particular kind of complexity or other things that we know are related to complexity, whether they are causes of complexity or impacts of it.

So, in the second report, we just came up with a list of measures as it turns out of software complexity. That will be expanded to more of systems complex-ly related, for example, to the avionics hardware as well as the software. In other words, we are trying to determine in what ways we might possibly we able to measure the complexity of what we have.

We started in the second phase by looking what is out there and collecting these metrics for software again. We will come up with a little bit, perhaps, more innovative or on the edge or out on a limb type of metrics. Then, in a later phase, we are going to reduce those numbers and bounce them against data to see which ones actually predict safety problems.

Suzanne: So, you are really going from a list of attributes, measuring a list of attributes to trying to find something that I'll just call, for simplicity's sake, an index that helps you to compare. Because the purpose of these measures often is to say, *Is this more or less complex than that?* So, even though the absolute number may not, or the absolute measure, may not be as accurate as we would like, if we can get something that helps you to compare things, that could still be useful.

Sarah: Actually, you have hit on something that is another interesting aspect. When you talk about more or less complex, we have to look at two fundamental kinds of complexity. One is what I'll call *the objective complexity*, which means, *How many pieces are there in the system and how are those systems, those pieces connected? Are they tightly connected? Is it something that's a fragile connection? Are they resilient in a way that they aren't going to mess each other up? What is the function and the nature of the system connections?*

The second kind I'm going to call subjective complexity, after some work by UK [United Kingdom] Professor Hillary Sillitto. Subjective complexity is more in terms of the impact it has on us. So, the first kind can be complexity in terms of what causes it. And, the subjective is, *How does that affect us?* So, it makes it difficult for us. It makes it so that we can't predict. It makes it so that we feel there is risk and all these kind of subjective things to it. The funny thing I've discovered when doing this work is from year to year, the objective complexity: *How many pieces you have and how tightly or in what convoluted manner are they connected?* That increases from year to year. Every year things get more complex, and I don't think anybody's going to disagree with that. Oddly enough, at the same time, the subject of complexity of things that were complex last year are less complex now. The way I best describe that is, when the Wright Brothers first flew their airplane, that was the most complex thing that ever went into the sky. It was the only thing that ever went into the sky that actually worked, and it had all these numbers and things that people didn't know about. They didn't know about air dynamics. They didn't know about fluid flow over the aircraft or how to control it or anything. Nowadays, of course, our planes are much, much more complex, but the Wright Brothers looks ridiculously simple. So, anything we have this year is going to look simpler next year, less complex next year. So, the subjective complexity of any given thing will go down with time.

I attribute that, in a way, to the definition that we came up with, and I don't have it exactly. But, complexity is the relationship of the number of things that cause problems to our ability to deal with them. Our ability to deal with them goes up as we develop good tools; as we develop, for example in the case of the Wright Brothers' airplane, understanding of the fluid dynamics of the atmosphere and of control theories. We have new tools like now we have computer-controlled avionics and things that reduce what the pilot has to deal with.

Suzanne: Right. Cognitive load on the pilot.

Sarah: So, we can deal with much larger complexity than we used to be able to.

Suzanne: That actual comparison--*Is this more complex than that?*--has a lot of contextual factors associated with it.

Sarah: I think so. I'm not inclined to do a *more or less*. Although, that is certainly easier to do than an absolute one because you can say that *A bottle of water is less complex than a pitcher of punch.* Because you've got all these different aspects of the punch. It has to be a certain temperature. Did you mix it properly? Is your sherbet melting as you are doing it? So, we can compare that more easily and that may end up being one of the tactics we choose to take.

On the other hand, the FAA wants to know, *How confident are we, based on its complexity attributes, how confident are we that it is safe? Is it 90 percent safe? Are we 50 percent confident that it is safe? After we go past some number, are we only 10 percent confident?*

I'm going to throw in the wrinkle that I haven't told them about yet that it might be only 50 percent safe this year, but maybe it'll be 90 percent safe after we have developed another way of predicting, or a new tool, or some modeling, or some virtual integration for example. It implies a way to mitigate and reduce the complexity.

You will never reduce, in my mind, the objective complexity. It is going to be worse next year than it is, but, you can reduce the effect of complexity in terms of our ability to deal with it.

Suzanne: So, the current state of the practice, from what I'm hearing you say, is that we have got a handle on objective complexity attributes. There are a lot of those that are understood. The subjective complexity, we have some things that we are looking at, but that seems like that is the area where you are going to be doing a lot more research to connect subjective and figure out what is our ability to deal with different kinds of objective complexities. Is that where we are at?

Sarah: Well, that makes a lot of sense, but I don't think I can do that. Even though there are measurements out there for different types of objective complexity, we don't have anything that—except for the McCabe complexity, which is within one particular program—we don't have anything that has proven to be measurable in a way that is consistently measured from case to case in different implementations.

Suzanne: In different contexts.

Sarah: So, I think there is a big step in coming up with something that is actually usable and comparable across different kinds of aircraft systems.

Suzanne: And, you are looking for something that is persistent.

Sarah: Something that will be true, yes, over time. In terms of subjective complexity, I think that is almost too far in the future. I mean, we're just starting to understand that that is important right now. I don't know. Maybe we will come up with something great. Stay tuned.

Suzanne: Well, we're not going to discover anything unless we start researching it, right?

Sarah: That is correct.

Suzanne: So, this area of research is really in its infancy in some ways. I know complexity has been studied for decades, but this aspect of trying to actually build insight into the confidence of *How complex is it?* and *How does that relate to other quality attributes, like safety?* I can imagine if you figure out how to predict safety, somebody's going to want you to predict security and other quality attributes in the future. So, you can be busy with this for quite a while.

Sarah: Absolutely.

Suzanne: So, you are really at the beginning of that kind of connection and trying to figure out what are the important aspects of subjective in connection to the objective complexity and building a taxonomy for objective complexity to work with.

Sarah: Maybe that is where we have to go. It would be great if we could figure something useful out of it.

Suzanne: So, people are still out there building systems for airlines that are software-intensive. Is there something they can do now, from your initial research, to help them do a better job of building their software systems?

Sasrah: Well, my thought about that at this point is that a literary and academic taxonomy isn't necessarily of use, but I think in one way it is. Because, when you are building a system that is getting more complex, I think just having a list of things to consider about how complex they are that you can then go and rate, even on a subjective scale of [1.] *Really, really a big problem*, [2.] *Might be a problem*, [3.] *Probably not a problem*, or [4.] *Don't even think about it*.

If you can just have that four-point scale on these things, then what you can do is take your *really, really big problems* and your *probably a problems* and study what risks they leave you with. And, then, *How can you assess the risk?* and *How can you mitigate it*?

So, if you take some of the more complex systems, science type of things, such as *How much chaos might there be in a system*? or *Who are the nodes and the edges and the independent actors*? Those are some ways I see right now that you can interpret your system, a lens you can view it through to determine where your risks might be.

Suzanne: And, then, we do have some newer tools, like the virtual integration work that Peter Feiler and his colleagues are working on, that actually give you some ways to test some of those ideas and think about them from a modeling viewpoint, which is very new in the last 10 years for us to be able to do that. So, there are some new avenues for actually using that insight that we didn't have before.

Sarah: In my wildest dreams, I would have…Do you know those tree finders that we used to have? They are a little book, and you would flip the pages. If the leaf had three lobes, you would go to page 86. If it had one lobe or is round or something, you would go over there.

I would really love to have some sort of complexity finder that you could do some sort of an internet quiz even on your system, and it would tell you, *Well, you are going to have problems here, there, and the other place*. There are such things that exist in the systems-of-systems world.

So, the kinds of experience that we are going to need in looking and dealing with complexity of software systems is not simply computer science. There is a lot of that icky, touchy-feely stuff that we tried to get away from when we became engineers in the first place that we are going to have to deal with and [we] have to learn how to do that. I think that systems engineers have learned, and they have been dealing with the people aspects to some extent. I mean, they are not anthropologists, and they are not psychologists, but they do have a requirement to understand the system into which the system they are building will fit. I think that is also the direction software engineers are going to be going, at least the bigger picture of software engineers rather than the coders.

Suzanne: So, you have two phases complete. What is the next phase for this research?

Sarah: We are trying to do more of a study of these potential metrics that we came up with. And as I mentioned, extend those to some of the system methods. More importantly, try to connect those to safety implications and see which ones are more indicative of safety issues.

Suzanne: OK, all right. Well thank you very much for joining us today. This is very interesting. This is research I have not had a chance to spend much time with, so I'm very excited about it. I want to thank you, and I want to thank our listeners for joining us today.

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.