Agile in the DoD: Tenth Principle

featuring Mary Ann Lapham and 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. Today, I am pleased to introduce to you to myself and Mary Ann Lapham. Welcome to our ongoing series exploring Agile principles and their application across the Department of Defense. In today's installment, we explore the tenth Agile principle: *Simplicity—the art of maximizing the amount of work not done—is essential*.

First, a little bit about myself and Mary Ann. Mary Ann's work focuses on supporting and improving the acquisition of software intensive systems. For Department of Defense programs, this means assisting and advising on software issues at the system or segment level. In addition, she leads research into software topics that are germane to acquisition of software-intensive systems, including the SEI's research in adoption of Agile and Lean methods in regulated settings like the DOD.

My research focuses on synthesizing effective technology transition and management practices from research and industry into effective techniques for use with Agile and Lean methods in regulated settings like the DOD. This is our tenth podcast exploring the real-world application of Agile principles in Department of Defense settings. If you like what you hear today, there are already nine podcasts in this series, and we will eventually have 12, one for each Agile principle behind the Agile manifesto. As a reminder, the four values and 12 principles of the Agile manifesto can be found at www.Agilemanifesto.org. Today, Mary Ann, let's talk about simplicity and the work not done. What are challenges to that goal in DoD Agile settings?

Mary Ann: You know, Suz, when you said the principle first, my mind immediately went, *What do you mean work not done?* It just seems like an oxymoron. *I have a project.*

Suzanne: We get paid for doing work, right?

Mary Ann: We get paid for doing work. *I have a project. I need all these things with the project. What do you mean you're not going to do some of the work that goes with it?* That was my immediate thought from a standard, traditional, acquisition perspective: *Excuse me? Work not done?* So, that is the first thing. Anybody that wants to use Agile principles is going to have to get past this. It doesn't mean not giving you what you are asking for. It means maximizing on your return on investment, and getting the value you need, and then determining if some of the bells and whistles aren't needed.

Suzanne: We are also talking about the difference between creating complex architectures and complex ways of solving a problem and looking at, *What is the simplest way*?

Often, when we say, *What is the simplest way to do something?* we actually stop having to do a lot of the work that goes along with implementing the complex way. So, the simplicity is not just about looking at getting value, it is also about reducing complexity. We are very good as engineers in figuring out lots of convoluted ways to make things work. So, this is really saying, *Don't go there if you don't need to*. I go back to the old Einstein quote, *Make everything as simple as possible; but no simpler*. That is what you are trying to achieve here: *figure out what is the simplest possible way of implementing something*.

It goes into the way you work. The simplest way of showing people what is happening is sometimes Post-it notes on a whiteboard. You don't have to use a complicated tool that you are going to have to buy licenses for and everything, if it is a small project. But sometimes that is not the simplest way you can do it. Sometimes you do need some of the other tooling. If I am in a distributed team, the simple whiteboard in my place is going to be complex to try and photograph and duplicate. So, I need to find another simple way to communicate with people. It is about that philosophy to me of, *What is the simplest way of doing things, but no simpler?*

Mary Ann: I agree with that. The other thing that, when you were talking, came to mind is, one of the other principles where you create a team, and you give them the license to go and solve the problem--collaboration and all that. When you challenge a team with, *Here is what I need you to solve. You have two, three weeks, depending how long your sprint is, to solve it*. They're going to very quickly discard the very complex, interesting, but not necessarily needed solutions. They are going to get to the most simple way to get what they need to get done, and still give the answer that is required.

Suzanne: We go back to some of the timings that are common in Agile settings of the two-week iteration, the three-week iteration. You very rarely see anything above four weeks. Part of that is to kind of force that simplicity and figure out what are the essential things that need to be done to realize the value of that set of requirements, of that set of items in the product backlog. So, all of these things, all these principlrs do interact with each other.

Mary Ann: Very much.

Suzanne: I want to talk about some examples that we have seen of enacting this idea of simplicity. One of the most successful programs that we have worked with in the Air Force has been through two complete re-architectings of their system in the last 10 years. Both of those times they were driven by some technology. The world of technology is changing. They had to go to service-oriented architectures. When we talked to them about what their goals were for those re-architectings, one of the goals was always *Make it simpler*. So, that concept, once you get that concept in your head, that is how you look at things: *How do I make this simpler for my user? How do I make this simpler for my developers, for my testers, for my certifiers? All of those things*.

Now you are not just looking at, *What is the technical simplicity*? I am also looking at the simplicity of the process. That is where we also get into maximizing the amount of work not done. Because if I am working shoulder-to-shoulder in a collaborative setting with someone, and I can talk to you face-to-face, there are some things I am not going to have to write down. Because, once this conversation is over, we are done with it. We have made a decision. We have moved on. If I have to go through three people to get approval on something, now I have got to document it so that everybody has the same understanding and all those sorts of things.

So, some things about Agile processes simplify and maximize the amount of work not done by moving you into closer collaboration and closer communication. That is a different way that you can get that simplicity-- look for simplicity in the process. There are Lean methods like value-stream mapping, where you look at, *How do I get the valuable aspects of this product out*? and *What are the steps that add value along the way?* to help you understand where you might be adding in steps into your process that are not adding value. All of these are things that play together in looking at both the complexity and the simplicity of your product and the complexity and simplicity of your process.

Mary Ann: Right. When you said the process part, I started thinking about work in process. Some of the more Kanban kinds of things. People will look at those now and determine, *Well I don't need to do all this stuff*. The simplest way is not to put everything into play at once, but to do it in groups.

Suzanne: To finish what we started.

Mary Ann: To finish what you start first, as opposed to dividing and conquering and having too many things going at once.

For those of us who have been around back in the day when you had machines that you would go out and poll, and every 10 seconds or whatever it was, they would poll the next port, and they

keep doing this. If you got too many ports, you would end up with a machine that was thrashing. Some of the more elegant solutions, although complex, end up with people thrashing when you don't need to go there. It's human nature to want to be…

Suzanne: So, this is really a mindset shift.

Mary Ann: It is a mindset shift, and it also….

Suzanne: Now there is a contracting aspect to this.

Mary Ann: Well there is a contracting aspect, but before we get to that, there is also the engineering mindset where you have got to make sure your engineers are trying to get the simple solution and get to the end first. Because they like to do…

Suzanne: Experiments.

Mary Ann: Complex experiments. Yes, you can call them experiments. It is fun to play. They are geeks to a certain degree, and there is nothing wrong with that. But they need to understand, *Where is the end, and how much time do we have to get there?*

Suzanne: *Is this the time to be experimenting?*

Mary Ann: Right.

Suzanne: Or, is this the time to be producing the value?

Mary Ann: Some of the programs that I have talked to recently, they actually allow a session that it is part of a hardening sprint. But, they don't use it to fix things with the program. They allow them to go out and play and do what-ifs. The only goal is they have to have something that is beneficial to the program at the end of the two-week sprints that can be demoed that will help them solve some problem they have run into.

Suzanne: There are some of the scaling techniques for Agile, like the scaled Agile framework actually. They call that a hardening innovation and planning sprint. And, that is starting to be adopted by some of the programs that we're seeing.

Mary Ann: OK. So, now that we have talked about the engineers, and how they are—well, I am one now.

Suzanne: Yes, you are.

Mary Ann: Geek on the forehead. We like to play, and we like to figure out new and different ways to do it. One thing you have to realize is simplicity is sometimes better, and we can get

done with that part. Now there is the contracting side. They [warranted contracting officers and some program management staff] aren't technical, typically, and they have very rigid ways of doing things.

Suzanne: Program managers are generally not as technical.

Mary Ann: Generally not. Although if you are lucky, and you get an engineer who has turned into a program manager, they understand the engineering side, and they can help you get to the simplest solution faster. But, they also have to deal with the actual contracting officers, and *How do we put this into play*? And, sometimes it is not simple.

Suzanne: This is where, from a contracting viewpoint, you have to know that if you are in an Agile setting, that there are going to be times when people come back to you and say, *We don't need to do this*. If you have constructed your contract in such a way that every detailed requirement must be executed, then you are going to actually be doing work that doesn't need to be done to provide value in the solution.

One of the recommendations that we have made, and that we have seen work well, is to really be careful about what is the contracting level of the requirements that you put on the contract. If they are at too low a level, you are going to end up doing work that is not needed; you are going to end up adding complexity to your solution. And, you are going to end up spending money that doesn't need to be spent.

Mary Ann: Right.

Suzanne: This is a real area where if you know you are going to be contracting for Agile, look very carefully at *What is the level of requirements that I am making contractually required to be implemented?*

Mary Ann: It is one of those situations where, too high of a level isn't enough to give you enough oversight and insight in what is going on.

Suzanne: And confidence.

Mary Ann: And confidence. And, too low of a level paints you into a corner where you are going to end up doing things that after you get down a certain point in your design and your architecture and your development, you will realize you don't need to do…

And, if it is already on contract, you either have to go through the lengthy contract change process, which is also time consuming and expensive, or you do the work that really isn't needed, but it is contractually required. You get into this infinite loop.

Suzanne: I will give you one example that we heard about. I won't name the program, but we do know of a program that actually hired a SETA [systems engineering and technical assistance] contractor. That is a contractor that provides services to the government. They were put in between the developer and the government program office, so that they could do the documentation that was required by the contract. Even though the program manager didn't necessarily believe that all that documentation was needed for the end solution, it was still contractually required and had to be produced. They didn't want the developers spending their time doing this. So, they actually hired someone that also understood the system to do this for them.

Mary Ann: You needed them up front, but as things evolved and technology changes, or the requirements from the field change, some of the stuff becomes OBE'd [overcome by events] but it is in a hard contract.

Suzanne: Overcome by events.

Mary Ann: Yes, overcome by events.

Suzanne: That is one of the things that we frequently see in longstanding programs especially. When you have a program that starts out with one idea of what technology is, in today's world the technology environment can change much more quickly than what is typical of our acquisition lifecycle.

Even though we probably have more to say, we have probably said enough on this one.

Let's talk about the next episode. Mary Ann and I are going to explore the eleventh Agile principle: the best architectures, requirements, and designs emerge from self-organizing teams. We have talked a little bit about this before, but we are going to talk some more about it in terms of the technical aspects of how self-organizing teams work.

Listings for papers, blog posts, and podcasts related to SEI research on Agile adoption in DoD can be found at http://sei.cmu.edu/acquisition/research/index.cfm.

If you would like more information about the SEI's recent publications in all areas of our work, you can download all of our technical reports and notes at http://resources.sei.cmu.edu/library/.

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.