Applying Agile in the Department of Defense
First in a Series

featuring Suzanne Miller and Mary Ann Lapham

Suzanne: Welcome to the SEI Podcast Series, a production of the Carnegie Mellon Software Engineering Institute. The SEI is a federally funded research and development center at Carnegie Mellon University in Pittsburgh, Pennsylvania. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts. My name is Suzanne Miller, and today, I'm pleased to introduce you to myself and Mary Ann Lapham. Instead of just interviewing today, I actually get to participate.

Mary Ann's work focuses on supporting and improving the acquisition of software-intensive systems in many different ways. For Department of Defense programs, this often means working with the program office to assist and advise on software issues at the system and/or segment level. And she performs and leads research in software topics that are germane to the acquisition of software-intensive systems. In this case, what we're talking about today, agile methods. She's been doing project management for over 35 years, so she's very versed in the traditional ways that people approach project management and software management. She's been leading the agile research in ASP, the Acquisition Support Program, since 2009.

My research in general focuses on synthesizing effective technology transition and management practices from research and industry into effective techniques for use in the acquisition of complex systems in the DoD and federal space. I've been working with Mary Ann since 2010 on the agile research as well. That's been my main focus in doing this.

When we talk about agile methods, the term "agile methods" derives from the Agile Manifesto, which is a document that's published on the web. If you go to www.agilemanifesto.org, you will find it. You will find the four values that are the basis of any of the methods that call themselves "agile." You will also find 12 principles that are the ways that you can tell if you are actually acting out the values.

One of the things that we found with DoD and federal clients is that these principles are a little bit new. Some of them feel good—they feel like they fit within the DoD culture—and some of them don't. We thought what we would do is have Mary Ann and I discuss what we've seen in relationship to each of these principles in DoD or federal settings. So, each of these podcasts in this series will talk about one principle, and then we'll discuss our experiences in dealing with that principle in the federal and DoD space. So Mary Ann, why don't you kick us off.

Mary Ann: Thank you, Suz. The first principle, for those who aren't familiar, is "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." Now all agile projects try to meet that principle. In the DoD some do, and some aren't quite so adept at it. I've seen a couple programs that actually do it quite well. They get the software out there fairly quickly, within 2 to 4 weeks, and it's ready to go. It's potentially shippable. As you may know, that's what you're supposed to have at the end of a sprint or iteration, for those who aren't familiar with the terminology.

The problem I see ~~is~~ for the guys in the field, if this is already a fielded system, or even if it's being developed; they can't deal with that frequent of a release. So, they lump them together, and then they send them out every 8 months, 9 months, 18 months, whatever it is that works for them from a deployment viewpoint.

Suzanne: So, even though we're producing valuable software as a program, we may not actually get to deliver it, in the way that commercial settings might be able to deliver it.

Mary Ann: Correct.

Suzanne: So, I've seen what we call a "sandbox" as one of the ways that we deal with that. So, in a sandbox setting, I will put each iteration's software into what we call "a sandbox area." Customers who volunteer to work with it—essentially to be beta testers-- get access to the sandbox, get access to the code early; theycan give us the feedback that we need, which is why this is really an important principle. That whole principle is about getting feedback. If I continuously and quickly deliver software—real, working software—out to people that are likely to use it, then I'm going to get better data about what works, what doesn't work, where my ideas and their ideas converge and diverge, right?

Mary Ann: Right.

Suzanne: So, that sandbox is one of the ways we've seen that you can get that feedback even if you can't do full delivery out to a complete fielded setting. Now, the other setting where this sort of delivery is a little weird in many of the DoD systems is embedded systems, right?

Mary Ann: Yes.

Suzanne: So, talk about that for a second.

Mary Ann: Well, with embedded systems, you're tied to whatever you're building. You're building an airplane or a ship or spacecraft, space vehicle, satellite kind-of-thing. Of course, you're not going to deliver software to the actual functioning entity (physical thing) until it's ready. You don't just have half a ship that you put out to sea.

So, that begs a different issue of "What do I do with the software that I'm getting out there?" Again, sometimes they use the sandbox-type idea; they don't call it a sandbox, but they'll have incremental drops, and they'll have a test environment, kind of a test bed, where they simulate a lot of the other systems so that they can actually check it out in a more operational environment. There's a lot of testing that goes on there, and again, they get the same feedback. So, it allows them to find integration issues early as opposed to waiting until what we would call the "big bang," when in more traditional systems, you don't do anything until the very end when everything is ready, and then you put it all together. So, if you can do this early stuff with the software, that helps them be ready for when they do the final integration with the ship, plane, whatever the embedded system.

Suzanne: One of the challenges I've seen with that is getting user involvement. You can get testers, you can get the systems engineers, you can sometimes get the program-office people, but the most value is usually seen if, in the case of say an airplane, you get a pilot who looks at how the navigation system is going to work and talks about, "Well, yeah, but when I'm doing this, you know, something else is going to be distracting me." The solution to that, I think, in many cases, is getting people more aware of what the benefit is, of having that early involvement,because it does take work for somebody to go out of their operational setting, come into the program office, look at the software, go to an iteration review, and play in the sandbox. The value to them is that they get so much more insight into what's coming, but they also get to provide that feedback. And then we get to have a better system.

There are other principles within the set that drive you to take that feedback and use it. So, this one is really saying, "Do the first thing," which is do the early and continuous delivery of business-valued software, which could be operational value in case of an operational system. It may be business value in terms of "bang for the buck," in terms of features in a new development. So, that's the other piece of this—that you're not just delivering software for the sake of, "Oh, I think it's cool to make this." You're delivering software that has value to the people that are paying for it, and the people that are using it.

Mary Ann: Value in the eyes of the end user. Now, one of the things a lot of the programs are trying to do, since they can't get access to the end user—after a while when the end user figures out that, "Wow, this is really something that will help me," then it's easier to get access to them. But, it takes time, and it's a different mode of operation. So, a lot of times, the developers will get what they call "surrogate users," people that used to be in that field, maybe they were pilots and have retired but are still involved in the field and keep up with it. So, they come in, and they

act as the initial reviewers of it. And, then they go talk to their buddies that are still on active duty, if it's for the military, and say, "So, I saw this great new thing and here's..." and they'll go, "Oh, yes." or "No, that's not how we do it anymore. We do this now." So then they can take that information back. That was probably over a cup of coffee as opposed to actual field operations, but it's still information that helps.

Suzanne: The other way to involve users is one you and I both got to participate in—there's a program in the Air Force that every couple of years has a user forum that's an actual two-day meeting. It's like a little mini-conference. They get around 300 people. They have 600-plus installations of their software worldwide. So all these people come together and review it and do training on all the upgrades.

There are cases in the DoD already where programs that are using agile methods as a base have figured out different ways of doing the early-and-continuous delivery and assuring that what they're giving out is valuable software, in some cases to a very diverse audience. A lot of times our audiences are much more targeted, but there are some that are pretty diverse.

We've been talking about release rhythms, what I would call "release cadence." Talk a little bit about "early and continuous" from the iterations viewpoint. So, in most agile methods, you've got a release that is something you are intending to deliver, more or less to an actual end user. Then, you've got the iterations in between that are much shorter than we are typically accustomed to thinking about.

Mary Ann: Well, typically what they do is they'll group capabilities, and maybe a capability will take five or six releases or iterations, excuse me

to get the full capability. At the end of each iteration, you'll have a piece of it, but only a piece. So, then you'll get all the capability. Then if it's say, six, nine months, and that's a time frame where it's reasonable to put it out, if it's an operational community, then they'll release it out sometimes as a beta, but sometimes an upgrade to the system, depending on what it's for. Now, the delivery is sometimes nine months too soon. Some of the larger organizations, they just get something in, and nine months later you're bringing them something new. They're just basically finishing the initial training on the one you gave them nine months ago. So some of them only want delivery every 18 months. In that case, you might have a couple of releases that go out at the same time. You'll have a release ready, and when the field's ready for it, you can send it out. In the meantime, you start working on your next release. You have to work with your operational community, or whoever your end users are, to determine what that cadence is. It depends on what it is you're delivering, who's using it, and how frequently they can support delivery. A lot of it is the training; some of it's the installation...

Suzanne: And let's not forget certification and accreditation.

Mary Ann: Yes. While a lot of testers are on board with Agile, some of the more operational testing organizations within the different military branches have their own way of doing things too. Sometimes the release may go to them first for their work and then out to the field.

Suzanne: So, when you get into thinking about iterations, if you're approaching agile, we've been seeing some really interesting things. Iterations in the commercial world can be as short as a week. About the longest you'll see out in commercial is 30 days as an iteration. Now, if you're accustomed to a normal big bang, you know, big design up front, you go, "Oh my gosh, how could I get anything real done in 30 days?"

I've got at least two cases I've seen now, that use the 30-day cadence for iterations, and they operate really differently from the ones that I've seen that operate on two-week iterations. There's something about the urgency of that two-week iteration that seems to do two things. One is you really do decide what's the minimally viable product, to get something real out in those two weeks. The second thing is, and sort of driven by that, is it makes you look at your backlog of features and make them more granular because sometimes you envision things in too big a blob. We actually had one case that we heard about from one of our programs where they couldn't figure out how to decompose this one feature. The most that they could decompose it would take you into a three-month iteration, which is very, very long. That's a release in most places, not an iteration. They couldn't figure out how to do it. So, the manager said, "Okay, I trust you guys. If you think it's going to take three months, let's go ahead and experiment with that." So, they created a three-month iteration...

Mary Ann: And, what happened?

Suzanne: Two months in, guess what? They're coming back to the manager saying, "Uh, this isn't working. We're not going to make the three months. We're going to need another month." At that point, he was a smart guy, he said, "Okay, waitLet's stop here. Let's see what we've got. Let's break this thing down." Now the good news is after working with this big thing for two months, at that point, they actually understood how to decompose it to where they could go back to the two-week iterations. But, that's a case where what he said afterwards is—you know, hindsight is 20/20—"anybody says that to me again, our first two-week's sprint's going to be figure out how to do this; because we're not going through that again." So, that was an interesting case of "We're going to go back to something more traditional," but it didn't work in any case.

One of the things I'm coming around to as a personal opinion is try to do the shorter iteration. Try to do the two-week iteration, even if you have to do things like a documentation sprint, that's separate from the software-delivery sprint. Even if together it still turns out to be 30 days, you'll break the paradigm of doing mini-waterfall where you're trying to do all the requirements for the sprint first and all that. You'll actually work more like we expect to work in an agile fashion,

where you're doing real software from day one of a sprint. To me, that's an interesting thing that we're finding in the field.

The other thing that relates to that is that if you're on that 30-day iteration—many programs have a monthly program review, where they go over everything—if your sprint cadences are on the same calendar as that, it doesn't really feel very different to the management. So, I actually think there's a thing about making it feel different by being on a two-week cadence. You emphasize the fact that this isn't quite the same as what it always was. That's an intuition thing. I don't have data for that. My intuition is that that difference is one of the things that can make your managers start treating you differently, not just treating you like a one-month version of the rest of the program.

Mary Ann: That might work. And I never really thought about that. One of the things I did notice though is if you allow it to go four weeks or even some I've seen have been up to six weeks—the longer you allow it—again, people don't see the change, and human beings tend to procrastinate. If I have four weeks, well I don't need to do that today. I can do it tomorrow. So, you don't get into the urgency that really makes, I think, the agile processes work, because people realize they have no time to waste; they really have to get to work and get down to business.

One of the other things I've seen too is something they call "hardening sprints," which also takes out some of that other stuff. When you're getting a release ready to go out in the field, there are all these extra things you have to do to make sure it's ready. You do those in what they call hardening sprints. So, it's a special sprint just for that prep stuff, finishing off, tidying up all the paperwork that needs to be done, whether it's documentation or a C&A package (certification & accreditation), or getting all the final approvals lined up and you have the package all ready to go. But you do that in a separate sprint so that it is different, you're treating it differently, and it doesn't get confused and intertwined with your agile software development piece.

Suzanne: And, you get a little separation of those...

Mary Ann: Yes, and the separation, I think, helps people change over to the new way of looking at things.

Suzanne: I think that's probably as much as we want to talk today about the things that we've seen in relation to this principle. I want to mention a couple of reports that relate to this. One is a report that's coming out in the second quarter of 2013 that is about what we call "Alternate Worlds." It is taking sort of familiar concepts from the traditional viewpoint and then talking about what is the agile concept that relates to that.

If there are terms that we used today that you weren't familiar with, this is a document that should be very helpful with that.

The other document I want to highlight is a technical note we published in 2012 on information assurance issues in relationship to Agile. And this idea of how do we get information assurance integrated into Agile so that that doesn't create a huge lag at the end of a release is one of the things that's dealt with in that paper. So, those are just a couple that relate to this topic that you may want to look at on our SEI website.

Mary Ann: Actually Suz, I think we changed the title "Alternate Worlds" to "Parallel Universes..."

Suzanne: Ah, okay.

Mary Ann: Either of those titles, I'm not sure which one it is right now to be honest, but if you can't find it in one, look for the other one, and you'll find it.

Suzanne: And Steve Palmquist is the primary author so that is the name you'll find.

Mary Ann: Yes, find it under Palmquist.

Suzanne: Mary Ann, thank you so much for joining me today. I love talking to you about this stuff. In the next episode, we will deal with the second principle. There are 12 altogether, so we will look forward to these conversations in the future. If you'd like more information about the SEI's recent research in general, there's lots of stuff besides just the agile work that we do.

You can download all of our technical reports and technical notes at sei.cmu.edu/library/reportspapers.cfm. This podcast is available on the SEI website at sei.cmu.edu/podcasts and on Carnegie Mellon University iTunesU site.

As always, if you have any questions, please don't hesitate to email us at info@sei.cmu.edu. Thank you.