SEI Webcast

Agile in Government: Go for Insight, Not Just Oversight
by Suz Miller

Shane McGraw: Hello, and welcome to today's SEI webcast "Agile in Government: Go for insight not just oversight." My name is Shane McGraw, the outreach team lead here at the SEI, and I'd like to thank you for attending. We want to make this discussion as interactive as possible, so we will address questions throughout today's talk. You can submit those questions through the YouTube chat area. We will get to as many as we can.

Our featured speaker today is Suz Miller. Suz is a principal researcher here at the SEI working in the continuous deployment of capability directorate. Her current research is focused on synthesizing effective technology transition and management practices from research and industry into effective techniques for use in the governance of programs adopting or contemplating adoption of Agile or Lean methods. Now, I'd like to turn it over to Suz Miller. Good afternoon, Suz. All yours.

Suz Miller: Thanks Shane. I want to welcome everyone. As you can see, I am working from home like many of you. And so, I want to thank all of the people that make this kind of a webcast possible when we're in this kind of situation. It's a little different than doing it in the studio. As a couple of things about today's talk, one is this is not going to be an Agile basics tutorial. I'm assuming that the people that are listening have a basic understanding of Agile principles and Lean principles, and we're going to be talking about things from the viewpoint of the government program office. So, we're talking about things when there is a contractor involved in actually doing the development. So, think about this as Agile in the acquisition context more than Agile in the development context. So, with that set of assumptions I'm going to go ahead and move forward.

And what we're going to talk about today is a little bit of a manifesto. If you're familiar with software manifesto for Agile, this is the manifesto idea for Agile acquisition. This is notional. This is not anything that's published anywhere in particular, but it's the synthesis of ideas from the SEI's more than ten years of work in this area. What are things to pay attention to when you're in a program office and you're looking to move from oversight to insight? We're also going to talk a little bit about what are some traditional things that we think about differently when we're going for insight and not just oversight.

Bottom line up front, you guys can read most of this, but the idea is I'm not saying we're getting rid of oversight. I'm not allowed to say that, and I would never say that. We have things we have to do to meet the regulatory and statutory requirements of a program. However, we need more than those oversight mechanisms if we're going to get insight. Things that are part of our normal acquisition like a contract data requirement list items, required events like PDRs and CDRs, are not always the best way to achieve insight on an ongoing basis, and Agile development settings promote transparency at a much lower level than what many of us are accustomed to working in. And if you are able to achieve that, there are some built in mechanisms for achieving ongoing insight, but they require proactive participation from the acquirer to be effective. And so, that's

one of the big messages is that this is a change not just for your contractor but also for the program office and the other government stakeholders.

So, these are closely coupled concepts, right? Just like verification and validation, you don't really generally do one without the other. We are talking about that same idea that we are looking at things that you have to do on a continual basis, but their goal is slightly different. The oversight goal is to make sure that we are giving evidence of the process that's executing as correct, that it's appropriate, things like that and making sure that the capability requirements are managed and that they are feasible and those sorts of ideas.

Traditional project management, when we're looking for insight, we are looking for some of the same things, but we are looking for some other things as well. So, we're looking for correct understanding, for example, of the requirements for the environment. So, it's not just the requirements as stated but looking more in a validation sense at what is changing in the environment that makes these requirements still be the right ones. We are looking very much for providing adequate user participation to support early validation activities. This is a big hallmark of Agile is that we have much more involvement of the end user community and other stakeholders early. A lot of our backend processes, we actually try to do what we call shift left to get them more involved early so that we can prevent rework later on.

We're looking-- we still need the evidence for oversight. So, we still need that, but we also need to periodically review the processes and work and make sure that they can evolve. One of the things that we know about processes as well as products is that the world changes, the requirements change, and we have to be ready for that.

Fundamentally, we are also looking at managing relationships, communication, and interoperability within the project and with other projects. So, the program office is taking a much more robust role in managing all of the interfaces of the people that are involved in the whole program. So, this is a little different than what we might have done before.

And how much you think you need insight determines a little bit on your worldview, and these are extremes. I'm not going to say that these are here, quite frankly, to be a little bit provocative, but the idea is that if you're looking for things that are related to compliance, then you assume things like change is an exception. You're trying to hold the contractor accountable for everything, and it tends to be an adversarial. Prove to me that you did what I wanted is kind of the phrase that you would look for.

When you're looking for collaboration, on the other hand, you recognize that compliance really hasn't worked very well in many of our complex settings. We're looking to be cooperative, collaborative, try and understand each other's perspectives and get the best result, and, fundamentally, we assume that change is constant. We know that if the ground is continually

shifting, then compliance is not really what we want to be going for. We want to go for understanding, and that's a shift that happens when we start thinking about things from an insight viewpoint versus and oversight viewpoint.

And so, that's the first place I want to stop for a second and ask if there are any questions or comments we want to address in the chat? Is there anything for me, Shane?

Shane McGraw: No specific questions, but we've got a worldwide audience. We've got a number of people that know you personally are giving their hellos to you. So, if you want to--

Suz Miller: Hello, everybody. All right, I will look at the chat afterwards in total, but while I'm speaking, I'm asking Shane to kind of manage that so that I don't get distracted. So, I hope you will all be able to tolerate that. All right so, I'm going to move on then. Back to some principles, I am going to sort of remind you of the Agile and Lean principles. And the Agile-- the manifesto for software development is something that we want to move towards these ideas. How do these ideas apply to system acquisition? And these are the-- and remember, the balance of these is what changes. We're not eliminating what's on the right-hand side. We're just saying choose the left-hand side when you can. Those are going to get you better outcomes, especially for your small teams.

So, what happens when we move to system acquisition? So, what we've looked at in synthesizing from our research from these last ten years are what are the things that are making a huge difference in the success of the programs that we're working with, and they come down to four areas, batch size, feedback approach, requirements expression and management, and this compliance versus insight mindset. And so, just to contrast the same way that sort of the software manifesto does, what's different from Agile and traditional in each of these topics?

So, in traditional, we tend to have a few large batch interactions. We have large documents that get delivered. We have large, week-long review cycles for technical reviews. We do things in very large batches, but there aren't that many of them, which means there's fewer opportunities for feedback. What we're going for in Agile is many small batch interactions. This immediately changes the staffing profile of a program office if I am going to be interacting once a month or once every two weeks versus maybe once every six months with my contracting organization.

Second area is feedback approach. Especially early in the program for system acquisition, we're primarily looking traditionally at documentation reviews. What does this architecture document look like? What did these requirements documents tell me? And again, they are large batch. So, this is a slog when you're in a program office and you're trying to understand everything that's going on. When we move to Agile, we are looking not just for those, and this, again, some of this we have to do some of, but we're doing demos and user feedback as a much more prominent method of getting feedback and understanding what is actually happening. We still have to do

documentation review. That's actually part of our required oversight, but that's not necessarily where we're looking for the insight. We're looking for things that are more related to implementation, even early on.

In terms of requirements expression and management, in traditional, I say single, and I know that there are updates that come along with large requirements documents, but the idea is we are going for one single finished requirements document to guide the entire implementation, and here we're not looking at that model at all. We're looking at this idea of a backlog with a prioritized list of items that are various expressions of our requirements, but we're doing this continuously. We are looking at the backlog on a continuous basis to understand what needs to change because of what we've learned, what needs to change because the threat environment has changed, what needs to change because the technology environment has changed.

If you look at some of the requirements that programs are dealing with right now, there are some software programs that really weren't considering how do we deal with people that are trying to access this data remotely. That's something that changed in March and is probably nothing that was ever in any of their requirements documents, but the world changed, and we need to have a way of understanding what that change means, and this continuous backlog refinement is a strategy for that.

And then down at the bottom, what's your mindset? Are we seeking insight, or are we seeking compliance? We have to achieve compliance, especially when we start thinking about things like certification, but if we don't go beyond that, then we are going to get caught off guard in some of these other areas that we're not going to be ready for. So, we're going to take each one of these and look at them in a little bit more detail as to what they might mean when you're moving into an Agile space. And I am going to concentrate, even though I've got kind of a two-column approach here, I'm mostly going to concentrate on the Agile side as we go through these.

After we go through-- remind you of the principles, so these are the principles that go along with this, and the main reason I pop these up is to remind us that these are small team focused, and most of our-- many of the programs that we work with are not small teams, and that is actually a difference is that we're not looking in this idea of a manifesto. We're not looking at specifically small teams. We're looking at much larger programs.

The other thing that I want to bring up is my shorthand for how do you scale Agile to a large system acquisition is you don't. You do Agile at the small team level, but you use Lean principles from Lean engineering and Lean product development to actually deal with some of the system level things that are much larger in scope than what you can deal with on a small team. These are the safe Lean/Agile principles that many people are using today to talk about how they're going to go about their system acquisition and system development, more than

acquisition, and don't forget these. I'm not going to go through these today, but these are important to kind of keep in your head when you're trying to deal with the system level.

So, the real question is how can we gain insight into contractor expression of the Agile and Lean principles. So, the principles that I just put up, even if you're not doing them because you're not developing, they are what should be guiding your contractors. The Agile principles should be guiding their small teams, and the Lean principles should be guiding their larger teams. So, how do we do that? And I will also say, as a caveat, this is a short presentation. So, we're really just going to hit the tip of the iceberg on these concepts today. Before we move into sort of the comparisons side by side, I want to stop again, see if there's any other questions that have come up.

Shane McGraw: We do have some questions rolling in, Suz. We've got a good one from Larry asking, "Can you discuss the importance of PMO and developer colocation on the insight--"

Suz Miller: Okay, very interesting questions as we are all working from home, and none of us are collocated. So, in a traditional setting-- a traditional-- in a pre-COVID setting, I'll put it that way, where you can have PMO and developer collocation, you get what the opportunity for what Alistair Cockburn calls osmotic communication where you hear things as you're walking down the hall that help you to understand what's going on or something that needs to be dealt with. You get a much richer context of communication when you can have that collocation. What we have learned since COVID hit is, if you can't have that, you have to find other ways of getting that cocktail or that osmotic communication. It is harder, and you have to work harder at both informal and formal communications, and you have to typically up your technology game. One of the things that we're seeing is Zoom for government as an example. That was something that probably really would have been slow rolled if we hadn't had this kind of a crisis where people cannot be collocated, but it's always beneficial if you can have that collocation, but if you can't, you still have to go for high bandwidth communication, both informal as well as formal. Any others?

Shane McGraw: One more from Elsa. We have more, but if we could just squeeze one more in this section to ask. "I identified a bit of chaos when there are several squads in big data projects touching various areas. The more people the more areas, the greater the chaos and, consequently, delays and budget affection. What do you recommend in these cases?"

Suz Miller: So, I'm going to recommend a book that I'm just finishing reading that what we're addressing here-- what you're addressing here is some leadership issues as well as some team dynamics issues. There's a book called "Extreme Ownership" that came out a couple years ago that actually deals with how do you deal with the communications paths among people touching different things. So, this is not a book about architecture. It's more about the dynamics of that. The other thing I will say is that when you have lots of people touching the same product, from a

product viewpoint, I want to have an understanding of what the guard rails are on the governance of that. So, from the viewpoint of who's allowed to touch it for what purpose, that's something that the stakeholders that are involved all need to understand and promote and advocate for and comply with if you don't want to have chaos. It is something that requires quite a bit of discipline to be very honest.

All right, shall we move on from here? All right so, when we go from large to small batch, what are some of the typical small batch behaviors that we're seeing that are different? We look for even small pieces being implemented because we are looking to learn from them, that the shift-- the reason that we go for early implementation is knowledge gathering and understanding what it is we don't know yet. And so, that is a big piece that helps us when we go into small batches.

We also start really mindfully using this idea of stop starting, start finishing that is if you've done Kanban or other techniques like that. Finish something so that you actually understand it and have the knowledge about it rather than having so many things in process that you actually don't know when you're going to get done, which goes on to the next thing which is you want to limit work in progress to enhance the flow through the system. This is a whole lecture on itself, but trust me, you do want to do that.

This is a provocative thing for many people that a hundred percent utilization of resources is recognized as limiting flow flexibility and work accomplishment. "What?" you say, "How can that possibly be?" It's a true statement. If my calendar is completely full from Monday to Friday, and something happens that needs my attention in between them, I have no room for it. If I leave slack in my calendar for there to be some times that are not scheduled, I actually can take care of more things. I don't disrupt the flow of the things I've committed. It takes discipline to do that. Small batches are one of the ways that you accomplish that, but you have to acknowledge that you're not going for a hundred percent utilization of resources.

One of the big benefits of small batch behavior is that you have a short time between when a defect is found and when it was created, assuming that you're going all the way through an integration and test cycle for the small batch. This makes it much easier to find the defect and move on and correct it, so it doesn't affect other parts of the system.

You have to have lots of integration happening if you have lots of small batches. That is work, but it also is something that really builds confidence in the systems evolution. So, we have this tendency from all this to build quality in because we can build it in in small batches. Okay? And I'll let you read the other side.

When we move to primarily demos and other mechanisms like demos for user feedback besides document review, first thing I want to say right off the bat is demo does not equal test, but demo does inform test. The feedback we get from testers that have adopted shifting left into

participating in demos is they learn more about the system, so they can actually create better test cases, and they start to understand where testing is very robust in the development side where it isn't, so they can fill in the gaps where the testing is not already robust.

Active participation in demos is for small pieces of functionality. Going back to the small batch. We are not trying to create these complex huge war game kind of demos. These are much smaller things, and so that allows us to have more interaction with people that actually can understand what's going on in the small things.

We get open and continuous feedback about the fact of and the meaning of progress, or, if we're not making progress, what's the meaning of that lack of progress. The info from demos is fed forward into testing. So, this becomes a source of data for our certification and our test staff to say, "Here's things we've already done, and here's what it looks like." This is something that really needs to use the concept in Agile, and it floats up all the way into the system level of a definition of done, and that definition of done for every small thing and all the larger things needs to include certification criteria whether it's cyber, DTOT, ATO's authority to-- I don't remember ATC, but it's in there. Oh, authority to go actually certify, authority to operate.

Participation on continuous integration teams by government staff is seen as a high priority. So, this is-- something that happens as a result of going in this direction is contractors and government start to understand that we need people that understand the very technical infrastructure details working on the integration teams, and that needs to be a transparent part of the process.

Next one is looking at how do we do requirements expression, and when we go to continuous backlog refinement, we see a mix of push and pull communication across the government contractor interface on this evolving refinement requirements. So, you want tools like Jira. That's just one that is used a lot. I'm not saying it's the best, but whatever facilitated workflow management tools you have, they need to be on the same platform. We need to have that ability for push and pull.

You want frequent face to face, high bandwidth. We just talked about that. Meetings so that we can keep the relationship going not just do the refinement tasks. This is one of the important things about doing continuous backlog refinement is sharing enough about context so that the government understands what are some of the constraints on the contractor side, the contractor understands constraints on the government side, and we understand how to work through those things together. That is also part of transparency that is going to help to build trust.

And we see these frequent small batch prioritizations give us this solid base of understanding the current state and the progress. So, this is what we get out of that kind of refinement.

Last one is compliance to insight. Some typical insight behaviors, don't immediately react negatively to bad news. It is information that is meant to help make a decision and typically a different decision than the direction we were going, and if you don't want to discourage people from giving you the latest news even if it's bad, you can't be punishing them for it. That's just not going to work, and you want as much informal handoff of information where it's feasible, where it's allowed. Obviously, there are places where you can't do that, but to the extent you can, informal handoffs are one of the ways that you help people to help you have the right information, make sure you have the right scope, all of those kinds of things.

And this also allows Agile events that are allowed to preserve their cadence. If I am continually interrupting my three month planning cadence, that's more of a compliance behavior. I've got some other data call out here I've got to deal with, so you guys have got to drop everything and do this. That's not really going to help you with the insight, and the trust, and the transparency. Allowing the Agile events to preserve their own cadence builds trust that we can rely on this planning cycle and execution cycle, and we can get our work done.

We also see lots of sharing of test and certification assets. So, as opposed to if you know the criteria, oh my god, you'll develop to it, and I'm going to lose my independence. That's an isolationist viewpoint that is a very compliance related viewpoint. You start looking at things from the viewpoint of you know the criteria, you will develop to it, and that's what I want because, if I've done a good job of creating my criteria, then you developing to that means it's going to fulfill that criteria, and we all want that. That's a mindset shift, and no, you're not going to-- I'm not going to tell you every single little thing, but I am going to try and make it easier for you to actually do your job in the way that I need for us to be able to certify.

We select carefully the measures, and they are visibly used to solve problems not punish, and that's an area that's a whole workshop unto itself is to how do you achieve that in a particular setting, but you just don't take a template of here's all the measures that I could use and just slap them on. You're really looking for insight into the things that are going to help me understand what this team is doing, and with everything, you're really kind of having this collaborative mindset that we're in this together, and we need to solve problems together, not just walk along the same path next to each other. All right, I' m going to stop at this point and see what other questions have come up.

Shane McGraw: So, we've got lots of great comments, which I'm going to stay away from because I know you said--

Suz Miller: I'll look at those afterwards. That will be fun.

Shane McGraw: Later, but like just some questions. Larry had if you were going to get into any tools, like collaboration tools specifically. They use Jira in--

Suz Miller: Right.

Shane McGraw: Do you have other suggestions for collaboration and insight there?

Suz Miller: So, good use of wikis, and so DI2E is used by many people. It has a wiki. It's got Jira, which is a workflow management tool. Any tool that gives you the ability to do what I call unmoderated group editing. So, Shane can get into the document at the same time I am. We can edit it, and it will merge the changes and give it to us present it to everyone who has access to it, same thing with Jira. There are some newer tools coming out that do that that are very nice for group meeting settings. Unfortunately, most of them are not FedRAMPed, and so they are not available for use in typical program settings inside the government firewalls.

And you know, more important than which tool it is is that across the government contractor interface you're using the same one. If at all possible, you want to be using the same platform and actually the same repository. You want everybody to have access in there. That is very different than how we've done things in the past. In the past, developer's environment, government environment have been very separate. We are moving to a much more collaborative transparent, and that's really what you want to go after. The places where that has been achieved are places that I see more success in achieving this kind of mindset. One more, or move on?

Shane McGraw: Let's do one more. What about any-- what about implementation of program risk management, going from oversight to insight?

Suz Miller: Okay, so we, again, compliance, right? We have the whole ARM process if you're in the Air Force, acquisition risk management process, and we have to pay attention to that. We cannot let go of that. However, one of the things that, when you're looking at mitigating risk-- so, identifying risks is one side. Mitigating risks, one of the opportunities with Agile is to look at your risks from the viewpoint of which of these risks are actually mitigated by me instituting some of these things that I'm talking about in terms of changes in the way that I do batches, in the way that I do transparency. Which of these risks would be ameliorated by having more of an Agile mindset and more of a collaborative government contractor approach to things? There will be some that would be this is the mitigation, and I have seen programs that actually, you'll see that in their risk briefings about how are we dealing with this risk. Well, we're dealing with it by having more transparency, and here's how that's happening and things like that. So, that would be my first suggestion.

There is a group of people-- there's a community out there that says that, by doing Agile, I am immediately mitigating risk, and I understand their viewpoint of why they say that, but we have to be a little more specific in our environments about what about Agile is it that is mitigating risk. Well, small batches allow me to learn faster, and I have a risk that we have a technology

that we're really not certain about. So, I need small batches so that I can iterate and understand what I don't know. So, those are the kinds of things that you would do.

All right, okay so, let's go ahead. So, this is the beginning of a conversation, and I want to take this farther and actually get to a set of principles that help us for acquisition in the same way that they help you go from sort of these high concepts in this sort of manifesto presentation to actually being able to understand more clearly how to implement some of these, but that's-- over time, we'll get there, but it's something that is going to take quite a bit of work.

All right, the things we still need to think about but differently, and this-- I'll reflect on a couple of the questions that we've talked about in terms of how those things-- how they play into some of these things. So, the things I'm going to talk about very briefly, technical reviews, requirements, systems engineering, contracting, measures, testing, and certification. These are one page summaries of an hour and a half to two hours and sometimes a whole day's worth of content of things that we've learned about implementing these in different program setting, and we actually do have ninety minute modules for each of these, and we are now offering them as part of what we're calling a virtual schoolhouse. So, I and my colleagues do the kind of thing we're doing today but with a very particular topic like systems engineering, and it's closed usually to a single program, but we can-- there are some options for that, but what we get is a lot more discussion than we can really facilitate, and it's more specific to a particular setting than what we can do in this kind of a webinar. So, the link is there, and Shane will give you more information about that after this briefing, but I just want you to know there's more than these one pagers that you're going to see now. I'm just kind of cutting them down because of time.

All right, technical reviews, so what's the switch? The switch is projective versus as-built. Our typical formal high ceremony things like a preliminary design review and critical design review in the past have been very large batch, and once they are done, the opportunity for learning what we didn't know or what was wrong about what we thought we knew is much reduced. And so, taking these down to incremental reviews that are on a much shorter cadence, very typical is three month cadence, if you're familiar with doing program increments a la SAFe, that's sort of a typical cadence, doing the things you would do on a small batch is one of the ways that you mitigate that and you get-- you can use the exit criteria for formal large batch reviews in these small batch events. We may still have to do a traditional system requirements review. Some of these things we may not be able to turn into small batch, but once you can start figuring out my road map of things I need to have more knowledge of, you start to get an idea of what my small batches should be. And programs that have used this, especially early on for risk reduction, they are doing a very-- their early increments are very focused on risk reduction, and that's really what you want, but they're doing it risk reduction through implementation. So, that's a big deal.

Requirements, all right, I've got my little thing there that says if it's in the backlog, it might get done. If it isn't in the backlog, it won't get done. This is the mantra of moving to a backlog

mentality. This is very daunting if what you think a backlog is the three thousand four hundred and fifty-two requirements that have been prioritized all at once at the lowest level of detail. That's not what we're talking about. We're talking about understanding that fixed intent and variable intent exist together, and that idea of fixed intent are the things we know we know about, and they tend to be at a higher level of requirements baseline so that we allow the learning to occur below that level. We have used road maps to guide what needs to be specified in more detail and when things need to be detailed. So, we're not talking about three thousand four hundred and fifty-two. We're talking closer to a couple hundred, and maybe fifty or sixty of those are really detailed for what we're going to do in the next three months, but the others are at a higher level of detail, and that allows us to have trade space so that we can reprioritize when the world changes, and it also allows us to focus on the highest risk, most valued items earlier.

Systems engineering, what happens when large batch systems engineering meets small batch Agile software development? We get what we call Agile at the bottom of the V, and I didn't reference it here, but we actually do have an Agile Minute about Agile at the bottom of the V and what is it that is a problem with that. And the problem with that is that if you make all the decisions about what we're going to do before you implement anything, then you're going to be in the position of having to go back and do quite a bit of rework once you figure out that what we thought we knew is not what's real, and implementation is when you find that out. So, systems engineering moving to small batch is a bit of a controversial topic. We have seen some instances of this, and I will say that one of the things that moves in this direction is people that have taken the digital engineering idea to heart and are using model-based systems engineering have an easier time with this concept than folks that are still using very requirements-based documentation.

Contracting, I've put up, this is from Alistair Cockburn's Crystal Clear, and what are the properties of successful Agile project teams, frequent delivery, reflective improvement, osmotic communication we mentioned, personal safety focus, sunshine and visibility. I love this one. No dark places in the project. How do we contract for these kinds of attributes? That's really the question, and it's not an easy question. When we go to help an organization to rewrite their contracting documents to be more Agile, I have a list of thirteen documents that are typical in our acquisition ecosystem that I look at to try and say, "Are these Agile aware, or are they going to disable your move towards Agile?" Things like your test and evaluation master plan, your systems engineering plan, how your DIDs are written, how your technical review criteria are set up. Some of them are obvious from what I've said already, but there's a whole bunch of them that you need to look at for Agile awareness, and that's because we've got lots of places where we tell people how to do things-- we tell people what to do. Sorry, we never tell them how to do things other than to follow this standard or that standard, but what is it that we want them to do, we have a lot of that what do we want you to do that isn't necessarily aware of how things shift when you move to small batch, when you're used to demo based user feedback, when you move to continuous backlog refinements. So, these are things that you have to pay attention to.

This area is changing more than some of the others I've talked about. The diagram I've got up here, I've got a reference to DAU because this is the adaptive acquisition framework, and depending on the size and context of your acquisition, you may actually have a lot more flexibility than you had in the past. Major capability acquisitions that are complex cyber-physical systems have still not been adapted in this way yet, but software intensive systems that are middle tier and defense business systems actually have more contracting flexibility than they have had in the past.

Measuring progress, one thing that doesn't change when you start looking at Agile is what's the way to think about measurement. You still have to think about all of these attributes that you're trying to measure something about, technology effectiveness, process performance, product and size stability, resources and costs etc., etc., etc. And the question then is what changes in the way we think about these things when we move to Agile. So, process performance, resource, cost, schedule, quality, we still need to do that, but what happens when we move to small batch? What happens when we move to we're not projecting what's going to happen and not using proxies like documentation to say this is how we're going to measure progress? We actually have real software built. How does that change how we look at this? And this is something that is not trivial to think about.

There are three areas that are a little different that you want to make sure get included when you are adapting your measures. Flow versus utilization, I introduced that topic a little bit. Value, how do we ascribe value to what we're doing and understand what is value added and non-value added, and then what's our lead time from concept to capability? You will hear people in the commercial world talk about concept to cash. Our lifecycle is concept to capability. So, how do we get that? And again, this is a whole-- if you haven't seen Will Hayes' webinar on Agile measurement, you want to take a look at that because that gives you an hour and a half of looking at that whole topic.

All right and testing, this is a build that I'm just going to build out, boom boom. You've got to shift left, and you've got to move the whole cycle of certification and testing earlier. As soon as I have something to test, I want to be testing and integrating it. That is a different mindset. That is a whole-- that's a whole day's worth of talking, but integrating the Agile cadence with DT and OT certification, and certification is definitely a challenge. Those folks have a very different funding model and work planning model, and we have to figure out how we are going to interact with that, much more to get to what was on the bottom cycle there. So, those are some areas that we want to think about.

And certification itself is multi-dimensional. We have cyber. We have air worthiness. We have nuclear surety. We have healthcare, HIPAA, all kinds of certifications. And so, understanding how they can be involved throughout a process, this is just a notional multilayer organization

with a large solution Agile release trained Agile team, and I've just sort of marked places where a certifier might be thinking about how can they be more engaged. We use this to help communicate with the certification community that there are opportunities in this way of working for them to actually do things that are important to them, but they have to understand it, and they have to be ready to actually do something about that.

All right so, I'll stop here again. What else have we got that we can look at? I just had a couple-- we've got--

Shane McGraw: We have a question from your friend Merlin asking, "Glad you recognize that Agile doesn't scale well. In your experience, do contractors/developers handle the system level aspects while the participating teams use Agile?"

Suz Miller: So, we actually have a technical note on systems engineering that was written-- in Agile, that was written a few years back, and we identified three patterns. One pattern is systems engineers that say, "You all do whatever you're going to do, but give me what I want when I want it." and that basically means give me a detailed requirement spec early and things like that that you might not do in a typical Agile setting. The second pattern is where system engineers act as team members on an Agile team, and they are kind of the liaison between the Agile software developers and the system teams, and they help them to sort of translate what it is that they need at different points. And then the third option is where systems engineering is actually operating in a Lean/Agile kind of team pace themselves where they're doing small batch iteration. So, if you think of-- the reference I would give to you is the "Lean Machine" is a book about Harley Davidson's hardware development and how they went to small batch Lean. And so, there are settings that do that.

When we wrote that TN, there was only one organization that we identified in pattern number three. I can-- I can't name them, but I can at least identify two others that I personally see now that are doing this, and so it is becoming something that is more real, and pattern two, where there's a more liaison kind of effect, is actually probably the most dominant pattern right now. Systems engineers are recognizing that just leaving Agile at the bottom of the V does not serve their needs any more than it does the software integration needs, and so they are interacting more positively. Anything else?

Shane McGraw: Yeah, a question from Mel asking, "For very large systems without a full architecture or some big picture, doing Agile early can cause the team to go down many blind alleys that cost the project budget and time. How do you avoid this?"

Suz Miller: So, this is where vision and road mapping really come into play and what-- so, there's the ideal state and the more pragmatic what you're probably going to have to deal with. The ideal state is you get people to recognize, that are responsible for that architecture, that we

need a skeleton to be able-- a skeleton that we think is going to be stable to be able to work from so that what we do is not going to violate the creation of an evolvable architecture. That's an ideal state. If I can't have that state where we actually work together on creating that road map, then you're going to want to be very, very clear about when you do things in the Agile space, what are the assumptions you're making and what are the effects if the architecture doesn't look this way. So, it's more work for the Agile team, but it's also a way to get attention that you guys want me to go fast. So, you want me to do software fast, and that's all great, but if I don't have this information, I have to make these assumptions, and if they're not right, that means rework. And so, you have to kind of-- what you're doing is leading upward and getting them information but helping them to see where there are places that you can't really move forward effectively without getting interaction with that larger architecture and the architecture team.

Shane McGraw: Just real quick before final thoughts. Can we get the links, Cathy is asking, to the systems engineering options that you were just discussing?

Suz Miller: Yes.

Shane McGraw: And then Jay chimed in, "The Agile software development teams and systems engineering teams she's referring to." Are those in the slides by any chance, or is that in that--

Suz Miller: The references are not in the slides, but I'll give them to you after the fact so that they'll be in the-- the Agile in systems engineering. Actually, for each of these topics there-- except certification, we do have some kind of a technical note or publication. So, I'll make sure that Shane gets those.

Shane McGraw: And I can send those out to everybody. Okay, thank you.

Suz Miller: Yes. Thank you. All right, so final thoughts and maybe a couple more opportunities for questions, we'll see. So, for program offices, this can be a really big shift because it means changes in skill profiles. I need people-- we talk about owning the technical baseline, but if you're going to be interacting on a two week or one month or three month basis on the technical evolution of the system, you've got to have some chops. And so, you've got to make sure people have the skills on the government side to be able to be effective in making decisions as things are moving forward. Also changes in staffing curves for the program office. I am likely to not have these okay, everybody on deck. We've got a month to review all these documents for PDR. It's more I need you continually involved at a smaller capacity, but I still need you there every two weeks or every month. And it changes the character of interactions with contractors and stakeholders, much more this is what I need, how can we make this happen collaboratively than do this now and do this then.

Also, the changes in batch size is something that has a lot of implications. So, if you think about sea drills and the process that goes along with that, that's a very formal process and the idea of having things as sea drills that are Agile work products may not be your best idea because that means I'm going to have to go through contractor, formal review, and government formal acceptance and review every two weeks or every month or every three months. Thinking about how that change in batch size affects the way that you organize your business is really something that is paramount if you're going to move in this direction.

Why would you bother? What we're seeing is, especially where we need certified products, we can get faster delivery to certification, and if we've been able to get certification and DTOT to shift left with us earlier, we get faster delivery of high value solutions to our war fighters and stakeholders, and that's really what we're after is to allow our-- the speed of relevance sort of phrase is the thing that we're trying to get to, and we are never going to be Netflix, who's producing, I don't know, two hundred releases a day or whatever ridiculous number it is. That's not who we are, but every five years is not the right pace either, and every two years is not the right pace either in many of our settings. So, we've got to figure out better ways of doing this, and the system acquisition is a big part of that.

So, I'm going to say thank you. I will take some more questions if we have some time, and there is my contact information for those of you that don't already know how to find me. So, that's it. What have you got for me, Shane?

Shane McGraw: Great, thank you very much, Suz. That was a great presentation. We're combing through the questions. I'm going to read a comment from Adam just to see if you have anything to add to it. Adam said, "When we help our clients, we often do a gap assessment against the controls of XYZ standard, root controls under an initiative and into small chunks for iterative improvement."

Suz Miller: Yes, so that gap analysis can take many forms. There are places where you're really looking at gap analysis with the ecosystem, and we do something similar. We've got a white paper on Agile readiness and fit analysis where we're looking at where are you in terms of different ecosystem elements, some of which I've talked about here. And then you also do need to look at how can I operate at the team level all the way up through the enterprise. And often, there's a disconnect in that middle tier from Agile to traditional, and bridging that when you have portfolio managers and people that are managing the higher levels of the enterprise that are staying traditional but they want the benefits of Agile concept capability, at the lower levels, you've got to figure out how to navigate that, and I won't say it's easy because it isn't.

Shane McGraw: All right, Suz, we're caught up in the queue there. Again, make sure you check out the chat because there--

Suz Miller: I will. I will.

Shane McGraw: People you know, and again thank you for a terrific presentation. We hope to do a couple more here in the future with you as well.

Suz Miller: I'm happy to do it.

Shane McGraw: Special reminder for everybody that today's event was archived, and it will be available once we stop the stream here at the same URL. So, if you found value in today's talk, please hit that like button below and share it with colleague as well. And just a reminder, our next webcast will be August 19th, and the topic will be OCTAVE FORTE with Brett Tucker and Matt Butkovic talking about some risk management there, and we'll make sure we email everybody a registration link. And if you have any questions form today's event, you can always email info@sei.cmu.edu. Suz, I'll give it to you for the sign off. Anything you want to add before we go?

Suz Miller: Thank you, and I hope that you are successful in moving your systems towards a more Agile/Lean method of doing things because it does help once we get going on it, but I won't be the one that says it's easy. It's maybe simple in terms of the concepts, but it's a lot of work. There's humans involved. Any time there's humans involved, it's work.
All right, done.

Shane McGraw: So, actually, if you don't mind, we're not done. I've got-- one snuck in the buzzer here from Herman, a quick question.

Suz Miller: All right.

Shane McGraw: So, he's asking, "So, will the U.S. services be releasing incremental J-O-R-D, JORDs, or O-R-D-Ss?"

Suz Miller: So, JCIDS is the organization that does those kinds of joint requirements, and that's one of the focus areas of the adaptive acquisition pathways in terms of a desire to make that a more Agile incremental way of doing things, but that was not the first element of where they've focused change. So, I would keep up with that adaptive acquisition pathways website and see what you see in terms of joint ORDS and things like that coming down the pike and seeing how they're doing with that. I will say that is a big lift, and it's going to take a while probably to see really results in that area.

Shane McGraw: Terrific, thank you very much for squeezing that in, Suz. Folks, that's all the time we have for today. Thanks again for everyone attending and look for an email about the archive soon. Thank you.