Scaling Agile Methods

featuring Will Hayes as Interviewed by Eileen Wrubel

Eileen Wrubel: Welcome to the SEI Podcast Series, a production of 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 at Carnegie Mellon University. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

My name is Eileen Wrubel. I am the technical lead for the SEI's Agile in Government program, which works to assist the federal government in adopting lean and Agile software engineering principles.

Today, I am pleased to introduce you to Will Hayes, a principal engineer on our team. Will often serves as an interviewer on this podcast series, but today he is here with me to talk about a recent technical note that we co-authored on *Scaling Agile Methods for Department of Defense Programs*.

Welcome, Will. Thank you for being here today.

Will Hayes: Great to be here, thanks.

Eileen: Can you tell us a little bit about yourself, your background, and how you wound up in the Agile work?

Will: So, in 26 years at the SEI, I have had the occasion to work on a number of different efforts, CMMI program, chief among them. I have had a lot of opportunity to interact with people using software engineering practices in the field. An opportunity came up for me to join Mary Ann Lapham's team to look at how Agile works in government. I jumped at the opportunity, and I have not looked back since.

Eileen: Thanks. When we are turning our attention to this work on scaling, let us lay a little bit of groundwork for the audience in terms of, *In order to scale up first we have to start from the basics.*

So, if the DoD program manager is Googling *Agile software development methods*, they are going to see a lot of things about small teams of five to seven people, operating on small batches of work, on two- to four-week cycles. As time progresses the scope of the work gets more and more granular. So, that is the basic foundation. Can you talk to me about what it means to scale upwards from Agile 101?

Will: The challenge that I think our work began with was a challenge levied to Mary Ann [Lapham] that said, *Can this collection of methods and techniques even work in DoD programs*? If you have heard only the 15-minute version of what Agile is, you can only have the image of the seven, plus or minus two people, working together, perhaps in a circle, updating a website that takes a week or two to update. Clearly that does not typify the kind of work that happens in DoD programs.

As we thought about scaling, our initial thinking was, *How do we make this thing that works in the small, work with a larger number of people, or with a larger amount of work to do?* As we dove into it and started to speak with the opinion leaders in this community, it became clear to us it is not merely about making something that is effective in the small be effective in the large. But, it is really about understanding, *What are the attributes of the way you do things*? Not just, *How do we implement a practice?*, but *What are the things that drive your choices that needs to be common across these different points on this scale* So there is just as much challenge doing it as small as there is doing it in the large; it is just the pattern that we have figured out for the small is one that is very well understood.

So, Agile, at the team level, is a fairly well understood concept. When we start to need to go to a longer time horizon, a broader scope of work that will necessarily involve more people, the kinds of things that then could potentially start to break are things that we have not necessarily encountered as we have thought about how to define Agile in the small. That is what makes this work really kind of neat.

Eileen: In applying Agile approaches on larger programs, it is not just a matter of adding more Agile teams and telling them to talk to each other and operate in the same way, but we actually have to manage the interfaces between and among the teams and technical product. We have got a much larger systems engineering problem than we would have with one team working on a small website.

Will: Yes, and the mention of systems engineering is really key here, I think. Because, if you get only the 15-minute explanation of Agile, you do not see architecture. You do not see disciplined engineering practices in there. Fifteen minutes is not enough, perhaps, to go in to that depth. But, when you look at how people actually do their work, Agile teams plan more frequently, Agile teams communicate more often about architecture and design issues. It [communication about architecture, design, and interactions] is not made visible when you are speaking only about a small team.

When we start to look at programs of the scale that we see in the Department of Defense, then the need for a longer-term vision for the architecture, a need for broader communication across different disciplines—not just different teams—of what the design decisions are, those become more visible and more prominent. Because failing to do that well costs you so much so fast at scale that, perhaps, when we are looking at only a small team working on a small product over a short period of time then a magnifying effect of those consequences may not be as visible, but they are still there.

Agile teams, even just a small team of people working on a small product, absolutely need to apply discipline and systems thinking to what they are doing. So, the level of rigor, the level of engineering discipline applied, is not a consequence of the lifecycle model chosen but really is something that needs to fit the need of the system that is going to operate in the environment in which that system operates. That is what dictates how far we go in these things. And so, what we are seeing is there is a great commonality across the programs that implement Agile well, at scale. They attend to these things well. Just too as we would expect to see and need to see with the kind of systems we are talking about in DoD.

Eileen: Can you talk about those kinds of characteristics that they are attending to really well?

Will: There is a variety of models for talking about design and architecture. There was a great paper,a number of papers that come out of the SEI on architecting and thinking about Agile in particular. In one, there was a contrast among different ways of dealing with architecture.

One humorously put, "YAGNI [you ain't going to need it]." So, there is some perhaps naïve view that we can fail to attend to architecture and still be successful. I am not sure I have ever seen something like that and certainly not in the DoD realm.

Another is that we do the big design upfront. That is often criticized as, *You are going to put your stake in the sand prematurely*, so you are going to lock into decisions that you will later want to revisit. Another is we do it as a cleanup—or a hardening sprint, if you will—afterwards. We go in and refactor and reconsider the design decisions so that we can have a clear architecture.

When we look at real organizations and people who are successful using these concepts, there is a mix of all of those. One of the nice ways that I have heard this talked about is, *You want to make some of these decisions at the last responsible moment*. That is, there is economic benefit to preserving your alternatives as long as you can that avoids locking in prematurely to a solution that may in the end not work out.

Eileen: Agile really emphasizes this shoulder-to-shoulder heavy collaboration between an end user or customer representative and the development team. What happens to that when we need to start scaling Agile methods to the size that we need for large defense programs?

Will: A really important challenge that [I learned] early in my education about what Agile is, I started to really ponder this one. I had the privilege of taking Scrum Master Training from Jeff Sutherland. I cornered him during one of the breaks and said, *You have got this product on a roll. It sounds great, looks like it would work fantastic on a small team, but if we are building new aircraft carrier or new fighter jet, how do we define that role. Can a single person understand enough to play the product owner role that you have described*? *How do we scale that across different teams?*

Because, what I have learned about in Scrum master school is really relating to managing a particular team. Jeff's insight was keen. He said, *You are likely to have to have multiple people involved. It may need to be a team of people that helps play that product owner role when you look at scale*.

When we look at other scaling frameworks, we tend to see that. We tend to see people recognizing the enterprise view of things. In Scott Ambler's work, he differentiates an enterprise coach from a team coach, and I think that reflects this kind of understanding. When we look at Scaled Agile Framework from Leffingwell and his colleagues, we see roles specifically called out that address some of these concerns.

There is a need to address this capability on the team. The way you do it is not a single pattern that is implemented. When we talk about *scaling the full range*, there are different ways the pattern plays out to make sure that we get systems engineering and other stakeholders involved in a timely fashion, providing data that is needed for the team to go forward or the teams to go forwards.

Eileen: So there is a shift in the leadership perspective that needs to happen in order for all of these stakeholder behaviors and relationships to change.

Will: Yes. Often because of the potential adverse consequences of failure and the number of dollars and the timeline and the number of personnel involved, not having a firm plan that is then realized throughout the program is viewed as a failure. One of the interesting things about Agile is you are planning to fail. You are planning to fail in small pieces, so that you can learn from that to go forward.

Now, when I say planning to fail, I am sure that is way too exaggerated for the kind of audience we are typically speaking to. But, really, by shortening your iterations, by allowing us to defer design and architecture decisions responsibly to avail ourselves of future opportunities without undermining the needs that people have for having a basis for development, by doing this, we are actually buying options.

Eileen: So, there is a shift in leadership philosophy, or leadership perspective, that program managers and program executives need to take when they are looking at applying these methods at the scale that we are talking about in defense programs.

Will: Yes. It is interesting that you bring in leadership to this conversation, because when we spoke with Jeff Sutherland about what scaling means, his response was, *Well, scaling Agile is about making leadership Agile. It is about helping the people who are responsible for the long-term viability of the enterprise understand what implications there are to the way they do their jobs.* So how do we make sure leaders are supplying or assuring the supplying of information that is needed to go forward? But, also, locking in on, *during this timeframe, this is the set of priorities, and we are not going change them.* By having a short enough iteration when priorities change, as they often do, leaders can say *OK, we can stand to wait the two weeks while this sprint finishes before we introduce a new feature that is absolutely needed [in] real time.*

Leadership's ability to accept the cadence. Leadership's ability to contribute to the discipline of freezing the backlog for the short time period that it needs to be frozen. That is really essential for this to do well. And Jeff Sutherland's comment about Agile at scale is really about Agile leadership. It is really apropos here.

Eileen: So, you have mentioned Jeff Sutherland and you have mentioned Scott Ambler. You actually had an opportunity to speak with the thought leaders behind the five primary scaling frameworks in the industry today as part of your research for this paper. I was wondering if you would tell me a little bit about that.

Will: It was really kind of a privilege. These are people who are very difficult to get on the phone, and each of them gave us a full hour of time. I think most of them would have been willing to go further, if we had asked. Craig Larman on large-scale Scrum was one of the interesting folks we spoke to, and his perspective largely is, *Do not make it bigger, do it smaller.*

When we talk to Steve Messinger on DSDM, his perspective was, *Scaling is just as much about taking the big work and making it smaller as it is about taking the infrastructure of a small team and making it accommodate a larger piece of work.* And Scott Ambler's appreciation of enterprise perspective, I think that is a key one. And Jeff Sutherland's quote about, *It is about making leadership Agile* is really good.

Dean Leffingwell's incorporation in the Scaled Agile Framework of Lean concepts, I think, is a really high point for the things that we have learned about. You will see in this paper, the overarching impression that I had from our interviews with those five individuals was, while on the internet we might see people squabbling with one another about *This framework is better than that framework; If you are not using my method, you are being irresponsible*. None of these five people had any sentiments like that. Each of them spoke well of others in the field. Each of them spoke about how people who were engaged in the kind of work, they are engaged in really trying to add value and trying to be of service to the community from which they came.

That was a very welcome discovery on our part, because I think when we look at any popular new wave of technology or methodology, we get a lot of advocacy, a lot of people championing, and often that championing comes about through criticism of what they perceive to be the competitor. None of the five people we spoke to seemed to view the other four as competitors of theirs. They each seemed to view their mission as advancing the state of understanding. That was really a cool thing; for the SEI to be able to tap that sentiment was really neat.

Eileen: Is it safe to characterize using Agile approaches almost as a risk reduction approach on a program where we are taking these smaller bites and failing fast, or learning fast as it were, so that, as good stewards of taxpayer dollars, we are exploring in small pieces and very quickly learning, we are filtering out what is not going to work, so that we can filter down to a path forward that is going to make the most sense?

Will: This is a really key point, I think, for people to be successful at using Agile methods in the kinds of environments that DoD and other highly regulated settings demand. If you think about the due diligence approach we take to protecting taxpayer investment, what we tend to do on very large programs is to get a full breadth of understanding for the scope of the work that is going to be done and to take that to some level of detail to give our stakeholders confidence that, a.), we have thought through the full range of considerations, and b.), we have thought down to enough detail that we know where the difficult parts are.

Unfortunately, the stages through which we go, with the way we have tended to do this in the past, have focused on relying on proxies for what will be developed later. So, we have a requirements specification that is comprehensive and reflects what we will someday design and build. Then we have a design specification that we want to make sure is a high-fidelity representation of the requirements, but it is still a proxy for what we will someday build.

So, our method of assessing and managing risk in traditional acquisition has been, *Let's understand the full waterfront*. Once we have confidence that the development organization understands the full waterfront, then we can allocate the next level of funding.

Agile kind of turns that on its head and says, *Instead of surveying the swamp, let us find the meanest alligator and slay it. Then we will go after the next one. We will chip away at risk much more directly than projecting that this is the requirement, projecting that this is the way to* *implement it in the design, and then hoping that the implementation resolves that uncertainty about what the system will do.*

If you go into the PDR [preliminary design review] or CDR [critical design review], instead of with the full comprehensive spec you go in with a roadmap that covers that full breadth but evidence that you have already implemented the trickiest parts, you are bringing alligator purses to the decision authority saying, *We have already killed the meanest alligator*.

The proposition for benefit that we are looking at in DoD is that this may be a more effective way of buying down risk, because you are getting from concept to capability before you decide that you have got your arms around it. Whereas using those proxies of a requirement spec, a design spec, and the cascading set of artifacts that [they] create. Until you get to the final system acceptance test, there are things that you have not really had a chance to make sure you have resolved the risk, you have resolved the technical challenge, the way the user wants to see it.

By understanding the full breadth of the issues and choosing the hardest parts and demonstrating them as capabilities, we are managing risks. And we are having PDRs and CDRs of a very different nature than in the past.

Eileen: So, in these very large programs, a lot of the upfront expenditure is that projective, that *to be*, type of work. When we use Agile approaches, that up-front expenditure is doing and producing and exploring the problems.

Will: Yes, the original incarnation of the waterfall lifecycle describes a very, kind of, intuitively logical approach to staging the work you do. First, we understand the scope. Then, we understand the architecture. Then, we understand the design. If you turn the page over and look at page two, it actually says, *This is an approach that is fraught with risk and ought not to be tried.*

Most of that paper actually explores how it is that we need to maintain an understanding. *What level documentation do we need?*, and *Who do we need to share that with in order to make sure that everybody is on the same page throughout the development?*

As programs have gotten longer and longer and larger and larger in scope, the ability to do that effectively has really been challenged. So, this idea of *Let us take the scariest piece and do it first*, of course, after we understand what the full range of capabilities we want to develop are, that really is seen as a superior approach. I think the original author of the waterfall article would probably agree more than disagree with the reasons that we are seeing in Agile.

Eileen: So at the end of the day, it is fair to say we can still get these benefits of these small batch approaches and these shorter cycle times, even on larger programs, but we would have to pay attention to how we go about doing that business and how we go about maintaining that collaboration, the automation, all of those things that go along with the 15-minute introduction to Agile that we see for a small team?

Will: In that light, one of the interesting things from Craig Larman, one of the things he said about the purpose of Agile, is to make it so that we can *turn* on a dime *for* a dime. There are automation opportunities. There are ways of partitioning the technical scope that allow us to preserve options without paying dearly for them and allow us to avoid prematurely committing to a particular solution. That balance is a really important thing.

Eileen: You mentioned decision-making at the last responsible moment. Can you tell us a little bit more about what that means?

Will: This is a really counter-intuitive thing for many organizations. There is a story I tell in training classes. If you are the pilot of a commercial airliner about to make a landing at a brand new international airport, it cannot be your job to specify the runway or to go down there and build it. It has to have been decided and built before you got there.

But, the passenger terminal: maybe that does not need to be completely finished yet. Maybe we can open the airport and start doing business and collecting revenue before that is all finished. Furthermore, if we think about the shops in the passenger terminal—the coffee-shop chain that wants to open three stores in your very large international passenger terminal—if you could present them with data on the prevalent foot traffic and show who is going to be in what part of the terminal at what time of day, you might actually be able to negotiate for, as an airport authority, a higher lease rate.

There may be profit in postponing locking in those leases, whereas with the runway, you do not have that opportunity. That has to be present for anything to start. There is a set of things that, if you defer them, there is profit to that. So, in DoD systems, we might think about areas where we know technology is presently evolving. If our technical strategy can accommodate alternatives and does not lock into a single version of that technology, there may be profit. Profit in terms of more efficient approach, a more effective system, a longer-lasting, a more robust system, a system that has more interoperability because of that technology change. Finding ways to intelligently plan for that variable—as opposed to upfront locking down to every single thing upfront as our expression of due diligence—there is an economic tradeoff there.

Furthermore, back to the airport analogy, if you are going to serve the public, there are obligations you have about the level of illumination in areas that are accessible by the public, the height of counters, the availability of seating, and various accommodations. Those exist and must be abided. As we think about architecture and design decisions in Agile, there is this confluence of different kinds of constraints. Some you want to leave open. Some are prerequisites to other decisions being made. And some are, *You have got to do it this way to be in business at all.* Being intelligent about identifying those, and giving yourself the freedom to make decisions with more information later, is a really important part about doing Agile well, when we are talking about the scale that we are talking about.

Eileen: I understand we are also going to be releasing a blog post about this paper shortly as well.

Will: Yes, you should be able to see that on our website pretty soon.

Eileen: Thanks, Will. I always enjoy the opportunity to talk with you. And thank you for joining us. To read the technical note that we published on this work, please visit resources.sei.cmu.edu. Click on the Browse by Author tab and find Will Hayes' name. That is spelled H-A-Y-E-S. Also, please know that we will provide links to these resources in our transcript.

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 do not hesitate to email us at info@sei.cmu.edu. Thank you.