The Fundamentals of Agile

featuring Tim Chick interviewed by Shane McGraw

Shane McGraw: 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. To learn more about the SEI, please visit our website at www.sei.cmu.edu. My name is Shane McGraw, and today I am pleased to introduce to you Tim Chick, a senior member of the technical staff at the Software Engineering Institute. Tim works on the Team Software Process (TSP) initiative where he is responsible for defining, developing, and transitioning into practice high-performance software and systems engineering practices based on the principles and concepts in TSP and CMMI. His work includes applied research, product-and-training development, education-and-training delivery, and consulting in the domains of software-engineering and systems-engineering process improvement. In today's podcast Tim will be discussing the fundamentals of agile, specifically what it means for an organization to be agile. Welcome, Tim.

Tim Chick: Thank you.

Shane: So let's start off by you giving us a little bit of background on what agile is. It's a buzzword you hear throughout the software community today. Can you tell us what agile is and what it means to be agile?

Tim: Well, at its core, agile really came from the Agile Manifesto, which was first written in 2001. A bunch of guys got together, I think they were on a vacation actually, and they started talking, and they came up with this manifesto and then they published it. It's a website. You can go out and see it. From that it's evolved slightly. Now there's basic agile principles. There's 12 of them. It's really focused on self-directed individuals, “teaming” environments. There's other things that they value that's really core to their principles, like early delivery, a major focus on customer satisfaction, customer importance. There are the products, there are the services that you do first in this whole iterative development concept.

Shane: Okay. So let's talk to the listeners. Tell us, what are some of the specific approaches? What are the names of some of these approaches that people could look for?

Tim: So, the two most popular agile approaches are really Scrum and XP [extreme programming]. Scrum is basically a planning and project-tracking technique used by teams. It also has a really key element in terms of managing customer expectations. It uses a concept called a backlog, for example. It's basically the customer comes in and says, "Here are the story points or the features that I want in my product. Here is the prioritization of what's most important to me." Based on that, the team takes that input, they pick the highest priority features that the customer wants. They build a plan to do that in terms of what they can accomplish within a set period of time or time box (usually no more than 6-to-8 weeks, usually closer to the 2-to-4 weeks). Then they execute. They build those features. At the end, what they want to have is a deliverable software product to the customer, which the customer could, if they chose to, actually start using immediately. At a minimum, it should be working code, if nothing else as a demo, to get more feedback, to re-prioritize. That concept is, "Are we on the right track? Are we really building you what you want as a customer?"

At the end of that cycle or iteration, they would have what's called a retrospective, which is to stop and say, "Okay how did that iteration go? What could we do to improve it? What else needs to be done?"

Another key factor of the retrospective is refactoring, another technique, which is actually kind of an XP technique. It's based on the architecture, based on the design, based on the features we've implemented to date, "What code really needs to be rewritten to be able to maintain the product as well as quickly be able to give the customer more features in the future?"

Shane: So what's an agile organization look like? Is it easy for large companies, small companies? Is it an easier fit for the size you are, the type of work you're doing, can we get into that a little bit? What does an organization look like that's implementing agile?

Tim: Well, it's all spectrums. If it's a really large organization, it is usually a subset, like a division or a team or a project trying to implement agile techniques. If it's a small company, it could be the way in which they do all their products, all their services. Really what it comes down to is most organizations that are really successful at agile are usually on the smaller side.

Shane: Is there any research documenting this or backing this information up? Is there anything out there that people can go and look up showing that it fits into a certain type of environment?

Tim: Capers Jones published a book called Software Engineering Best Practices. I think it was released last year actually. In that [book], he actually rates a lot—I mean goes through hundreds of software practices, methodologies. Out of that, he actually ranked the top four in three categories: small, medium, and large applications. Kind of a way to think about what's a small, it's really an application that can be built by a team of probably about less than 10 people over several months.

A medium would be either one team over maybe a year or so, or maybe it's a couple teams over several months. A large application would probably be multiple teams over multiple years. That's kind of how he grouped them in those three categories. He defined them in terms of function points. I don't know if everyone is familiar with function points, so I like trying to translate that into people size. In that he basically ranked agile as number one in small, but agile really didn't make the top-four list at all in large applications. And he ranked agile number two in the medium.

Barry Boehm also wrote a book several years ago where he talked about the criteria. He had, basically, five key points saying, "Well, should I use traditional, plain-driven methods, or should I use agile methods?" And, based on those five criteria—for example, what is the criticality of the application (Would lives be lost or is it a convenience-type application)? A convenience application, he says, is more agile-based, while a life-critical application would be better matched for traditional methods. The other extreme on the axis is where more plain-driven techniques would be appropriate. He had four other criteria. So those are the types of things that you can look at that support that statement.

Shane: Very good. Now one thing I know is that organizations are interested in how to select the right agile method for their environment. How do they go about that?

Tim: So, I said Barry Boehm had the five criteria. I like to try simplify it to three. The three basic criteria are

1. *Is my organization ready to change, any type of change?* How hard and fast are we with our current status quo and approaches to doing things (If they are open to change, if they are open to ideas and not just in words but also in actions)? You know, any type of change—whether it's agile, whether it's CMMI, whether it's just a change in the building office space or whatever it is—it takes resources. It takes time. It takes effort. When you want to change your approach to doing business such as developing software, you need multiple levels of sponsorship. You need executive sponsors. You need the team leaders, the first-line managers need to support this change. You need developers willing to give it a shot and become efficient at the new methods and the new approaches. So that's one criteria.

2. *The other criteria is "What am I building?* What is my product that I have and that I need?" So for that one, the question is, "What are the characteristics of my product? Is it life critical? Will someone die if I have a defect in my product? Or, will they just be inconvenienced?" That really dictates which methodology or technique I should want to use.

Another example would be, "Who are my customers, my stakeholders?" If I have a really diverse stakeholder group, and they don't really agree on, "What is the top priority? What are the key features of my product?" I really probably need a traditional systems-engineer or requirements-

analyst type of approach for giving my team that feedback in determining, "What do I fit in this iteration? What do I fit in this time box?"

So that's really looking at, "Are some of the different agile techniques appropriate for my product that I'm building as well as my customers' expectations and needs?" Based on that, I can pick. Maybe it's a combination of some very agile techniques as well as with some traditional techniques—that marriage of methodologies, which is really what most organizations do.

3. *And the third and final criteria, really, is my organizational culture.* For example, pair programming, which is an extreme programming XP technique. If my organization really has a lot of interpersonal, collaborative type of issues (they would have a hard time with two people sitting together without arguing and really collaboratively writing the same code, using one keyboard, at one time). I really do have to deal with the collaborative issues before I would try to use the extreme programming technique of pair programming. So, those really are the three major criteria one should use in trying to figure out, "What are the right methods for me, for this project, for this organization?"

Shane: Right. Now, like anything else when companies are implementing something, a lot seem to struggle with implementing agile consistently through the organization. So how do you implement agile? Once it's implemented, how do you maintain consistency?

Tim: So, some people I've talked to, they really love agile. They love the techniques. And it's working really well for their team, for their project, but they are really having a hard time getting other projects in the organization to be just as successful as they are. That really is the key.

Other organizations I see that are doing agile, they like it, but they're really struggling with maintaining it. People come, people go. So, to fix or to address some of those concerns, some of those issues, on the TSP team we developed a planning framework, which is agile. It really is in alignment with the manifesto, as well as with the 12 agile principles, instead of the scrum technique, which really uses some pseudo measures. It uses features. It uses velocity. The planning framework that the TSP uses, it actually uses more precise measures like task time, schedule, dates, as well as size of your products. With that consistency of a measurement framework, I can now communicate across teams, across projects.

Where a lot of agile teams really struggle, where one team unit that's pretty stable, the team members stay on that same team over a decent period of time. They really gel in terms of their definitions of what a feature is or what a story is, but each team gets a slightly modified definition or interpretation of what that is. That really impacts your ability to estimate and to really build your commitments with your customer. So, your velocity really takes major changes every time you change a member of your team. That's one issue that a lot of organizations [struggle with] when they want to do it across the board.

The other one is in the size of the application that they're trying to apply the principles, right? With a larger application, it's a more complicated application—especially when you start talking about system-of-systems or multiple applications having to work together that are actually being developed in parallel with each other. For those, quality really becomes important. Regardless of small, medium, or large applications, agile pretty much says, "The most successful measure is working software." That's true. That's the goal of all software developers: to build a product that works, that meets the customer's needs.

But, as you get larger, that medium and that large approach, you really need some more leading indicators to quality. And so inspections, reviews—they really become key. You want to do those types of techniques on all of your products, right? Of anything—whether it's a document, whatever it is, your architecture, your designs—to help remove those defects as quickly as possible so you get that working product at the end, whenever you do do that integration for the customer—whether it's at every iteration, at the end of the year, whatever it is that meets that customer's needs.

So, in order to do that, TSP created this quality framework, which is collect a little bit of data about those defects, when are they injected? what is the impact of those defects as well as emphasizing the need to do early-on quality measures, such as reviews and inspections, right? So, once you have a focus on quality, which needs to be a little bit more than just working software for large or medium applications. And you have a more precise measurement for articulating where I stand on this project: am I ahead of schedule? am I behind schedule? do I need more people. All those types of things you do when you'remanaging a project. Well with just those four key measures: time, size, defects, and schedule performance, I now actually have a measurement framework that's pretty simplistic. But it allows me to answer a lot of questions—not just from the planning and quality perspective, but also, how is my process performing overall? Did I make the right selection between the traditional methods and the agile methods? and if I didn't, my data should tell me that, right?

It's that external looking in at myself, right? Because we get so busy, as humans into our day-to-day work, we really have a hard time objectively evaluating ourselves. And data is really good for helping us face reality, right? And so, with that and still every iteration, I stop. Just like in scrum, I do a retrospective with that little bit of data it says, "okay, this isn't quite working. Let's try this other technique, whether it's an agile technique or traditional technique. Let's see if this helps us, let's test that out one or two cycles. What does our data say?" And then we can either keep it because it's an improvement. We can get rid of it if it makes us worse, or if it's just a change for the sake of change, but really shows no benefit, we can pick and choose. Did we like doing it more than what we did before? If we like what we did before, there's no impact. Well, let's just pick that other technique, because it was easier and it was more fun. And so, that's really what teams have to do. And you have to be a power team to do that as an organization. By giving

them a consistent framework to make those decisions across the organization is really key to allowing those agile techniques where it's really knowledge worker, you know, collaborative decision making and environment. A lot of trust has to be there, right? And some of that consistency and core framework. And as long as everyone is following those frameworks, they can now communicate and they can start building that trust in terms of ability to make their commitments and actually articulate where they stand, right? Ahead or behind schedule? What's the quality of my product? Why did I pick with these methods verses other methods, right? They'll have data to help them support those decisions and those discussions with executives and leaders, which can also help maintain the sponsorship, maintain the need for—and be able to articulate the value of doing things the way they're doing it.

Shane: So, very good. For the folks or listeners out there that may be interested in TSP, if you could give them one site where are you sending them to? Where are you pointing users to go to to learn about TSP, if they want more information?

Tim: Well, there's two sites. Well, if you go to the SEI's website, so www.sei.cmu.edu, if you just do slash TSP, you can go to TSP's homepage. There you'll find some technical reports, things of that nature. If you want to hear from actual users of the TSP technology, I would actually go to the SEI's website. But after the edu, it would be slash TSP symposium, and go to the proceedings tab on there. Basically, there's five years' worth of presentations. You'll hear what Microsoft felt about it, Intuit, DoD, Hill Air force Base. So, there's just a huge variety. And I just think that's more powerful than any technical report.

Shane: Thank you for joining us. This recording and a downloadable are available at sei.cmu.edu. An audio file of this podcast, along with transcript, will also be available on the SEI website at www.sei.cmu.edu/podcasts. As always, if you have any questions, please don't hesitate to email us at info@sei.cmu.edu.