How a Disciplined Process Enhances & Enables Agility
featuring **Bill Nichols** interviewed by Shane McGraw

Shane McGraw: Welcome to the SEI's 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 in the SEI website at sei.cmu.edu/podcasts. My name is Shane McGraw, and today I am pleased to introduce you to Bill Nichols, a senior researcher in the SEI's Team Software Process Program. Bill's work at the SEI focuses on software development process management using the Team Software Process or TSP. Today Bill and I will be discussing how a discipline process enables and enhances agility. Welcome Bill.

Bill Nichols: Thank you.

Shane: First question, what does it mean to be Agile?

Bill: Well, it depends on who you ask. That sort of really is a catch-all supplied to a number of different methods, but they share certain common beliefs and values. Typically, people who believe themselves to be Agile, think that you'll get the best results if you focus on things like empowered teams, collaboration with your stakeholders, not doing unnecessary work, and getting frequent feedback.

Agiliests hate the term process because they use the word somewhat differently than we do. But if you think about process it's doing something repeatedly with some discipline, and it’s doing something to achieve an end. That's typically what the Agilests are trying to do. They're focusing on how you do the work, but they're focused on a principles level rather than a strict-rules level. Now, I've made that list pretty short by being very general, and that gets us kind of removed from being directly actionable. So, let's try to be more specific.

Bill Cont. What do the Agilests really do? In a recent Association for Computing Machinery communication, Laurie Williams published a couple of surveys she did. Now the most common response on "What are Agile principles and are they valuable, which ones are the most valuable" were because Agile teams choose among software development practices to be Agile they should choose practices that are in line with Agile principles. It sort of sounds like it's begging the question. We understand from that that if the teams are choosing their own methods then the teams have to be involved. Okay, that's important. If they don't have a specific set of practices then that comes out from that survey too.

So, what really are the principles? Laurie followed that up with some lists of principles, and had them rate them. It turns out there wasn't a lot of separation between a lot of the different statements, but the ones that stood out were

- **Highest priority is to satisfy the customer with early and continuous delivery of valuable software**. So, they are very focused on getting product out. Working software is the primary measure of progress. So, that's their primary feedback.
- **Deliver software frequently**. They're really talking a lot about deliver, deliver, deliver.
- **Build products around motivated individuals**. Now we're starting to get into the importance of individual and team motivations.
- **Give them the environment and support they need.**
- **Then at regular intervals, reflect**. You have some beginnings here of process improvement. What is the reflection for? It's how to become more effective, tune and adjust your behavior.

An assumption that maybe doesn't come out of that top five list, at least clearly, is the importance of feedback for replanning, that is continuously adjusting your plan.

Now, it's a practical response where you have imperfect information and perfect requirements. Conditions are changing. You have maybe process variation. You have to make an initial plan, but you have to collect data and use that data to feedback and change your plan accordingly, you know, adjust reality.

Now there are a couple of themes that come out of this too. I already talked about the focus on delivery of software. That's their objective, deliver software and deliver it continuously. There is a lot of focus on people, how you motivate people, how you get people engaged in the work. Third, improve the process. That's what reflection is for. So, those are their principles. What do they actually do?

Bill Cont. The most commonly used practices were

- short iterations, typically 30 days or less
- continuous integration
- defining "done." What does "done" look like? What are the criteria for "done?"
- automated tests, typically automated integration tests and automated unit tests.

At a practices level, you see a lot of focus on process and specific techniques. You know there are a lot of different techniques you can choose from, but these are the ones that get the most attention. Some of the lower-rated practices were things like co-located teams, coding standards, planning poker, stabilization, iterations, code inspections, burn-down charts, and lots of individual practices, some get used more than others. The key to people considering themselves Agile is are they choosing practices to support their principles rather than the specific practices they're actually using.

Shane: What are some of the Agile methods? You hear different terms. What are the more popular ones or the ones you most typically work with?

Bill: Well, the ones that I see most often are typically Scrum and that's an overall process, framework that has a planning phase, short iterations, and a reflection. Scrum doesn't focus as much on the specific practices, so that's an exercise to the reader.

Shane: You've explained what Agile is and what it means to be Agile. What are some of the problems in Agile development that organizations face?

Bill: The biggest problem today is software is getting bigger and bigger. The big question naturally is how do you scale? How do you make this work for larger organizations, for larger project sizes? Things that work within a small team, with people that can talk face-to-face, don't necessarily scale when you go to bigger projects. The ways we run small companies are very different than the ways we run big companies. The way we manage a small community or a homeowner's association is different than the way we manage a city or a state.

I think part of the problem that Agile has with the large-scale projects is that there's still a lot of variability, and that comes back to the discipline. When you have a lot of variability in the individual parts of the project, the big project can suffer from that weakest link, as in the weakest link of the chain. A large project can be taken off the rails if just one or two parts of it aren't functioning well. I think that's certainly part of the challenge in Agile, getting more consistent behavior out of all of the individual tanks.

Shane: So you have to reward the team and not necessarily the individual.

Bill: That's a big part of Agile. Agile focuses very much on the team rather than the individual, and part of our critique from a TSP perspective is you really have to start focusing more on the individuals because the individuals are the people who make up the team. Individuals, just as teams, can affect an overall project and program. Individuals really can affect the teams, and we'll have some things to say about that.

Now, going back to the theme of the discipline, Jeff Sutherland, who is one of the co-inventors of Scrum, had a 10-year retrospective. What are the challenges that face Agile, and a lot of them are very familiar, they're very familiar to us doing TSP, things like

- lack of a ready-to-develop backlog. The backlog or sets of requirements or what you did. Is this really ready to begin development.
- lack of "done" at the end of a sprint. You get to the end of a sprint and you might not have done your testing. You might not have really done your continuous integration. Have you done code reviews? Maybe not, is that really done? You can have too much emphasis on meeting that milestone, that end-of-sprint deadline and forget about some of those important things in your process.
- What about the daily impediments? Fred Brooks said, "How does a project end up two years behind schedule?" One day at a time. Even on a 30-day schedule if you weren't managing your work day-to-day, you can find yourself behind. The daily Scrum is supposed to avoid that, but they don't always work.
- Tolerating defects. If you get a bug list at the end of your sprint, if you take the time to fix them, you might miss your date. It could be very tempting to do the easy thing and that is, just tolerate the defects, call it done and ship the code.

At the bigger level, you have lack of organizational change agents. Most of the Agile methods, being very team focused, haven't yet evolved to organizational change or training to people in Scrum to be leaders of organizational change. That has the fall through that you lack the agility in the overall organization.

Now you see some technical issues here, but from our perspective a lot of this comes back to the lack of discipline. I mean we have these sets of things that we know we should be doing, we understand we should be doing. We know how to do them, but we still don't always do them. We do it most of the time, but sometimes we don't. When we don't, there often can be consequences that might not show up until later.

Shane: How do you motivate them to do these processes or stay disciplined. Do you have some benefits or some suggestions?

Bill: Well, the typical Agile approach is to use peer pressure. TSP has some additional mechanisms and that is we're very focused in TSP on measurement, process, and holding each other accountable for following the process.

One of the things that sets TSP apart is the concept of the role manager. The role managers are members of the team who each take responsibility for some aspect of the team management. If a team is going to be self-managed the team has to manage itself. Different individuals will take responsibility for being the conscience of the team with regard to different aspects. You might be the conscience of the team to make sure are all these things going through tests? Have you really done these inspections? Have you checked the results of the inspection? Were they done effectively? So, we have another level where the individuals will be the conscience of the team and working with each other to help insure the process is really used.

Shane: The next question, you kind of touch on it there, is in what other ways does TSP work with Agile?

Bill: Well, there are several things that we can say about that. I think there's some points of real commonality and some areas where we depart, but I think that those departures really reinforce the commonality.

We tend to be driven in TSP very much by defining the process and measuring. There are some reasons for this. We think that precise definitions, precise measures are the most effective and most meaningful form of feedback. For those to be meaningful you have to have a context in which those measures have some real meaning, which means you have to be doing something, reasonably repeatedly; that means following your process. With that in mind, let's go back to some of the compatibilities.

Typically Agile values people over processes and tools. TSP focuses on the people for the simple reason that it's the people who implement the process. It's the people who use the tools and the people who implement the technology. Plenty of research over the years has shown you have to focus on technology and tools and how they enable people to do their jobs. You can't just replace them in technology. It's creative work. You have to provide an environment and the tools to help the people be creative. So TSP is necessarily focused on how to work with people.

Now along with that we create this data-rich environment that provides many sources of feedback. It could be the rate at which work is being done. It could be the effectiveness of inspections or how much time you're spending in different types of activities. We try to measure and provide meaningful feedback at different points in the process. We're not just getting infrequent feedback. When we do these reflections at the end of a development cycle, it's based on data. It's based on data and real analysis. We have an opinion, we always ask well what does the data show? Do the data support that opinion?

Shane: Does this help technical debt versus helping them decide what can be put off until another date?

Bill: You make intelligent decisions based on what the data says, what does history say, and what are going to be the consequences? If we don't do these tests right now, if we don't fix these bugs right now, what are the consequences? I have seen organizations that quite literally have not produced any new code in a year or more, because they are now fixing all the defects from the field. That's a terrible place to be. If you manage the work you should be doing ongoing and do it the right way; that is the fastest way to get your job done.

Shane: Okay, last question, you know we've touched on this a little bit with a previous response, but many would say that process and discipline inhibit creativity. How would you respond to that?

Bill: Quite the reverse. In almost any domain, in any creative field—be it art, literature, or software—the most productive and the most creative people are almost invariably the most disciplined. Now it doesn't mean you rigidly follow a set of rules, but you know what the principles are. You know what things you can't get away with skipping, and you do the right things, and you follow that discipline. Let me talk a bit about the discipline of a process. I'll take an entirely different domain just to drive down this point. In music composition composers have known for years that it's easier if you do melody first, melody before lyrics. You hear that all the time. How do you know this is true? Everyone has kids who make up lyrics to the birthday song. It's easy. You have this structure that's defined a rhythm, a meter, a form of where the emphasis should go, and they can make up the lyrics. That structure that they've been provided, the hard part is making the melody. They can now unlock the creativity and you end up being more creative and more productive by following a simple process.

Guess what, it's the same thing in software. If you do good requirements and good design, you can now unlock the kind of creativity you need when you do the development. Doing things in a certain order makes you more productive and more creative.

There are a number of things where the discipline in TSP applies directly to software engineering, time management. People can't be creative when they're under time pressure. We don't do our best creative work. You have to manage your time. The greatest artists from Michelangelo to Picasso, they were very diligent in managing their creative time. They set aside time to be creative. How do you do that? Well one of the things is you have to set aside the time to prepare to be creative, whether its choosing the right block of marble or preparing the canvases. You have to set aside time that if you don't plan for it, it doesn't happen and it'll take away from your creative time.

We use the disciplined work habits to separate the creative from the routine. Being creative is demanding. It's tiring. It's a lot of work. In software the design activity is where we really express most of our creativity. If you try to spend too much time designing, you'll stop being creative. Yet we see a lot of developers who try to do their designing while they're doing the coding and you end up with a less overall productive approach. If you do your design and use real design notations, which are richer expressions of creativity than your programming languages, you can create an excellent framework so now the coding is less demanding. You can actually do more total work, and you can be more creative. You have to have standards.

An artist, for example, would want to know “Is this block of marble going to be adequate for my purposes?” Are there flaws that make it unusable? That's the problem we face with requirements. Are these requirements ready? What does a useable requirement look like? You need a standard. Similarly if you're doing a code inspection, you have to know is this thing really ready to be going into code inspection. Are these modules based on the data we see, based on the inspections and the tests, and are these really ready to integrate? If you don't have those standards in place, then you're going to be spending a lot of your time doing rework.

So the technical excellence has to be defined with a set of standards. What does excellence really look like? What is good enough? And that really gets us to the last part, and I touched on it earlier. Most software projects spend about half their time in tests. I don't mean just the unit test or the test driven development. I mean final tests with integrations, systems tests, acceptance tests.

Shane: Systems ready to go?

Bill: Yes, this is to make a final shippable product they spend about half their total time in testing. That's typical. There are plenty that do less. The typical is 40 to 50 percent just in that phase. And typical products spend about half their total cost and effort in maintenance. Maintenance means minor enhancements typically and bug fixes. Well, testing is not a very creative process. It's engaging. You have to solve problems, but you aren't being creative. You're just fixing the product. So, you've already got half the total product life cycle in maintenance. Of what's left, you've got half of that in testing. What's left to be creative? If you want to be creative break out of this trap, build it right the first time, and give yourself more time to do the creative work that we really want to do.

Our benchmarks tell us that if you follow it a disciplined process, a quality process, you can get that testing time, that late testing time somewhere between 10 to 20 percent of your total project. That's a lot of found time.

Shane: Excellent answer, Bill. Thank you very much for joining us today. Can you give the listeners some a list of publications where they can go for a deeper dive into your research?

Bill: I won't try to list all of the publications. What I would recommend is go to the TSP website on the SEI www.sei.cmu.edu/tsp/. There you'll find a link to the SEI library and the TSP Symposium. The TSP Symposium has a number of articles and some papers going back several years, some that have been written by us, many of them by our user community. They'll tell you what their experiences have been.

Shane: Excellent. This podcast and the SEI Podcast Series is also available on CMU's iTtunes U website. As always, if you have any questions, please don't hesitate to email us at info @sei.cmu.edu. Thank you for joining us today.