Adapting the PSP to Incorporate Verified Design by Contract

featuring Bill Nichols interviewed by Suzanne Miller

Suzanne Miller: Welcome to the SEI Podcast Series, a production of the Carnegie Mellon Software Engineering Institute. The SEI is a federally funded research and development center at Carnegie Mellon University in Pittsburgh, Pennsylvania. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

My name is Suzanne Miller. I am a principal researcher here at the SEI, and today I am pleased to introduce you to Bill Nichols who has actually spoken with us before. Bill is a senior researcher on the SEI's Team Software Process team. His work at the SEI focuses on software development process management using this Team Software Process, also known as TSP. Today we are here to discuss his latest technical report, an *Adaptation of the Personal Software Process that Incorporates Verified Design by Contract*. Welcome, Bill.

William "Bill" Nichols: Well, thank you. I'm glad to be here.

Suzanne: First question. For those of our listeners—I'm sure there aren't very many anymore—who aren't familiar with the Personal Software Process or the Team Software Process please, give us a brief overview of those.

Bill: Essentially, the Personal Software Process is a way of measuring software development work with some very simple measures: size of the product you're producing like lines of code, the activities or phases of development as you progress through from design to coding to testing, and measures of defects. *What are the things that lead to rework, that make a product wrong, so I have to go back and do it over*?

So, TSP defines the process. Because of the measures, you can go back and evaluate *how is the process working*? *Is it an effective process*? *If it's not an effective process—if it's not meeting your goals and I make a change—what did I change, and what were the effects of that change?* So, it's a very scientific engineering approach to evaluating the effectiveness of your software. Team Software Process takes these same principles and applies them in a team environment.

Suzanne: So, the Personal Software Process is all at the individual level, right? You're working within a team, but you're looking at your own measures. It's really just for your personal use to help you understand how to build your product better. The Team Software Process is really leveraging all that knowledge that an individual gets about their own process, so that you can make the overall product of the team better and better as you go along through the process.

Bill: Exactly. Team Software Process leverages the skills and the techniques that individual engineers practice in the Personal Software Process, and it brings it into a team environment, so we can produce superior quality products on time, on budget.

Suzanne: I know of at least a couple of projects that have had spectacular successes with this. There are lots of other reports that the listeners can look at for information on those.

Now you're starting to look at combining this engineering-centric process with another type, another aspect of software development called *formal methods*. So, why don't you tell us a little bit about formal methods, and in particular, how this *Design by Contract* might work within TSP and what made you attracted to design by contract as a formal method?

Bill: Well, let's look at this from a couple of perspectives. First of all, the PSP and the formal methods are, in a sense, decoupled. What we're doing with the PSP, the Personal Software Process, is using the framework and the instrumentation that the Personal Software Process brings to evaluate how does the designer, in this case, the Verified Design by Contract, really affect the results. Does it improve your results in some measurable way?

Measurable improvements would be: *I could do it faster. I could do it cheaper. I could do it with higher quality. I could measure that by having fewer defects in my testing phase*. So, PSP is a measurement framework that allows us to do this sort of scientific experiment, if you will. We can do this in part because we have a large body of people who have taken the PSP, using the standard PSP process scripts, and that's essentially our control. So, you have this built-in control group.

Suzanne: You're a lucky guy because most of us that do research in the field do not have control groups to compare our experiment with.

Bill: You see we actually have a couple of levels of control with the PSP. We have this *control group*, which has taken the course. We have hundreds of points of data. For some versions of PSP, we have a couple of thousand points of data on what is the distribution of performance among this random group of people. But, we also have the control in that because we are using a well-defined process. The process provides a basis for really understanding what did you do differently. So, we have that level of control, very precise, on what are we doing differently. We have this control group. *What is our base-line result?*

Suzanne: So, you can actually expect that the variation and performance actually relates to a variation in what's been done in the process.

Bill: Yes. We can get a lot of confidence that the differences in the results, if they're statistically significant, have this causal mechanism because we have a lot of control over what we introduce differently. We can look and see from the data that it was actually applied differently, that they spent some time using this process, that they were actually using this different process. And, we can look at the statistical differences between the two different groups: the treatment and the non-treatment if you will. It really is very much an opportunity because we have this great technology to do real experiments, real experiments in software engineering.

Suzanne: The formal method that you've chosen—Verified Design by Contract—say a little bit about that, so people can understand what's different about that.

Bill: Verified Design by Contract is a technique that has been proposed by Bertrand Meyer for quite a while. There are some recent papers that he's done on some experiments.

Design by Contract basically is [the idea that] components have a certain expectation of how they're going to perform and how they will behave. You are essentially setting up a contract. *This component is going to behave in a certain way. It's going to take certain inputs and give me certain outputs.* Now, that's very compatible with things like test-driven development where you're doing your tests early. In TSP/PSP, we've always talked about develop some of your test cases early. But, just developing test cases is not design. That's a factor of design. It's an aspect of design, but it's not the "whole-ality." It's not the complete part of design.

What we've done with the Verified Design by Contract is made some modifications to the PSP script to put that work in particular phases with particular outcomes, so that we can measure how much effort has gone into this Verified Design by Contract phase, to set up the contract requirements, to do some of the verification, the checking that the design is complete and that the design is correct.

We can now look later in the process and try to do some correlations with the types of time we're spending in later process phases, especially, the rework phases of tests. *Are we finding these defects now before we get into tests? Can we run these tests with fewer defects because we're doing this particular type of design, the Design by Contract?*

Suzanne: So, the hallmark of formal methods in general—and the Design by Contract has this also—is that you're using mathematical constructs to prove correctness.

Bill: Yes, and this is kind of a light-weight proof of correctness. It's attractive to developers because it's very compatible with the kinds of things they're familiar with. Yes, it's a provable, formal method, but it's also fairly lightweight because the kinds of things you do in Design by

Contract is you specify what the interfaces are going to be; you specify the particular behavior, and the constraints on the modules. It's the kind of thing that sounds like very normal, good programming behavior.

Suzanne: Yes, that's what it looked like when I read that.

Bill: It's very accessible to the developers.

Suzanne: So, that's the contrast with other formal methods. This appeared to me as something that I said, *Oh, okay. I could've done this back in the days when I actually coded.*

Bill: Exactly. We could've done this or we might have done something similar.

Suzanne: It's a little more complete.

Bill: It's a little more complete.

Suzanne: And, it's got some particular areas where you must give information and find out information about the product. So, you're actually using this to get a little bit better information for the software developer to actually complete the design in some ways.

Bill: Exactly. In the second week of PSP, we teach a few types of design methods. Designs are most effective if you have some way of verifying the design formally. There are different levels of formality, different types of design. You might have pseudo-code. You might have module decompositions. Adding just a little bit of formality sometimes can have a lot of leverage.

So, the hypothesis here would be that adding the Verified Design by Contract will not add a substantial amount of time to the design phase; that it will be some minimal amount of additional effort, but that it will lead to less rework, so that you will have no more total development effort for having put in the design. And, you'll get a lower level of defects coming out. So, quality is free. You'll put in this quality phase early. You'll do the right thing. You'll build it right the first time. It won't take you any longer to get it through code complete. When you're done, it's going to be a better product.

Suzanne: So, are there particular kinds of software defects that this technique is good at finding in comparison to some of the other non-formal design methods that you recommend in PSP or other formal methods?

Bill: Well, there are a number of types of defects that we typically do find in design. Now, Design by Contract is probably most likely to find—and this is a hypothesis. This is something we'll have to verify—but we use the IBM orthogonal defect classification scheme or something that was similar. It looked like that at one time.

They've moved on. Ours is kind of fixed, but things like functional defects are likely to be found with this kind of approach because you're looking very carefully at *How should the modules behave?* So, you hope that you find these functional defects before they get in to test. We should be good at finding various types of interface defects because now you're talking about, *What are the inputs? What are the outputs?*

We have some other techniques in design that might help with that but this is a little more formal. So, hypothesis would be that we may have fewer interface defects. And the interface defects are actually very common. It's very easy to have the wrong type of data passed or out-of-range data.

Suzanne: And so, some of the behavioral things that you specify would help you to identify when you're going outside of those boundaries before you actually get into the coding?

Bill: Exactly given this input I expect this output. In the simplest case, we've typically given people simple test cases. *Here is the input. Here is the output. Now, we're making this a little more formal. You can start to think about is this in terms of ranges.*

One of the powers of design is that you don't just do a specific instance. You aren't just thinking about a specific test case. You can start to expand the test case into ranges of behavior. So, just thinking about this from a design standpoint, you can hypothetically, on paper, think through, intellectually understand a number of different test cases, how a number of different test cases might behave. Then you might execute a few test cases as you're sampling to verify.

Suzanne: Okay. All right. This research is interesting from another viewpoint. That is that it actually is a collaboration between the SEI and some South American researchers that you found? They found you? Tell us about how that came to be and how that worked.

Bill: We've been working with Diego Vallespir at Universidad de la Republica in Uruguay. He is an authorized PSP instructor. He spent some time here as a visiting scientist. He has some students working with him at local universities—at the Universidad de la Republica and at Universidad ORT, which is nearby.

They've been doing some innovative work on experiments in software engineering. They like to do software engineering experiments. So, the concept here was the students have taken the PSP and recognized what you did. You don't have many opportunities to do really good experiments in software engineering. Well, here we have this ready-made control group. We can instrument the PSP and plug in the Design by Contract in a very sensible way. Now, we can run some experiments and compare this to the baseline data.

Now, the work that Diego did with us in the past year—and we just had another student complete some PSP analysis—they have a lot of experience with working with our PSP data. We did a couple of papers on the defect types that were being found in PSP data—differences between the defects in design and the difference with coding. So, this is very well aligned with some of the technical research they've been doing with our PSP data. It's a natural extension.

Suzanne: So, what's the next step with this research? You've got the scripts modified. You've got a proposal essentially of how this could work. And you've got some students and professors that are interested in experimenting. Have you set those experiments up? What comes next for us?

Bill: That is the next step. The next step is to actually perform the experiments and start looking at the data. If they are promising, [we plan to] see if the data can be replicated on a large enough scale to be convincing.

Suzanne: Well, that is going to be something to look forward to. As I said, I'm always excited when we can get data like this because it is very difficult when you're trying to do software engineering research in the field where people are actually trying to do real work. Those are not always compatible, so this is a really great example of getting past that and doing it with some young students who are going to sort of take things like this forward.

Bill: Well, it's really satisfying to work with the students who have the motivation to go ahead and do this sort of research.

Suzanne: So, this is the first I'm aware of, of formally introducing a new design technology in concert with PSP like this. Are there other types of methods that you and the PSP team are thinking about introducing beyond Verified Design by Contract?

Bill: Well, we've been thinking that this is the kind of direction we would like to see PSP go in the future as sort of an instrument for applying future research, so that you can actually test out new techniques and now models. Verified Design by Contract seems like a very good fit.

There are some other possibilities, which I can't think of precisely at the moment. But, there are some others that some researchers in Portugal have proposed. There's no reason that you can't experiment with different types of approaches. Essentially, when you're trying the scientific method, you want to try to have some control over the changes you're introducing. Well, PSP is perfect because we have a defined process. PSP is ideal because we have this large baseline of control data essentially.

Suzanne: So, if you're a professor out there with a new design idea, and you want to test it out, you may want to get in touch with Bill Nichols and see how it works with the PSP.

Bill, thank you very much for joining us today and talking about this. I'm very excited about this work. I know this is a new stage in the PSP research that all of us at the SEI are looking forward to. So, we look forward to your results in those areas.

If you would like more information about Bill's latest technical report as well as all of the SEI's technical reports and notes, please visit sei.cmu.edu/library/reportspapers.cfm. For more information on the research Bill's team is doing on the Personal Software Process and the Team Software Process, please visit www.sei.cmu.edu/tsp/ and click on the research tab.

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, don't hesitate to e-mail us at info@sei.cmu.edu. Thank you very much.