The Evolution of a Science Project

featuring Bill Novak and Andy Moore Interviewed by Jay Marchetti

---

Jay Marchetti: 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 Jay Marchetti, and today I'm pleased to introduce to you Bill Novak and Andy Moore. Bill works in the SEI's Acquisition Support Program and has more than 20 years' experience with real-time embedded software and electronics product development; business development; research and development in software engineering; and product management in defense contracting, commercial, and small-company environments. Hi, Bill.

Bill Novak: Hi Jay, great to be here.

Jay Marchetti: Andy is a senior member of the CERT technical staff where he explores ways to improve the security, survivability, and resiliency of enterprise systems through insider threat and defense modeling, incident processing and analysis, and architecture engineering and analysis. Before joining the SEI in 2000, he worked for the Naval Research Laboratory investigating high-assurance system development methods for the Navy. Welcome, Andy.

Andy Moore: Hi Jay, thanks for having us.

Jay: In today's podcast, Bill and Andy will be giving us some highlights of their recent technical report, The Evolution of a Science Project, a preliminary system dynamics model of a recurring software-reliant acquisition behavior. Bill, let's begin by having you tell us about your research described in this technical report and the real-world problem it addresses. Can you begin by explaining what a science project is in the context of your work with the Department of Defense?

Bill: Absolutely. Well, first of all, a science project is a kind of colloquialism, a common phrase that you hear to describe the development of some kind of advanced prototype, typically to demonstrate the proof of concept of a new idea. What can happen sometimes with a successful science project, if you will, is that it will demonstrate a really important new capability, and people will see it: people in the government, military, etc.—whatever the interested organizations are—and may find it really fills an important need, so important that they may want to actually put it into use, sometimes in the field as soon as possible. So, the system can go out into the field and get some initial testing there and become almost wildly successful.

The only downside to this is the fact that what is becoming successful is actually a prototype that was developed perhaps rather rapidly, rather than a full production system that's actually ready for production-quality use. So, as the program tries to deal with its own success in the field and now with the requests for all sorts of new features that are coming in to it, at the same time it's trying to grow up into being a full acquisition program. This is where things can become sticky, because you're starting with a prototype that was developed perhaps in haste, perhaps not with all the documentation architecture and sorts of things that should be there. As it tries to grow up into a major acquisition program, there can be real shortfalls between what's needed for that, and what exists. A decision has to be made at some point, "Well, can we continue to build upon this foundation of a prototype or not?" That's essentially the dilemma that occurs. When programs try to keep building on a somewhat unsound foundation that wasn't intended to be the basis for a full system, that's where we can get into performance problems that can plague the program from there on out.

Jay: Can you tell us what is system dynamics modeling, and how does it relate to your research?

Andy: Sure. I'll take that one, Jay. System dynamics has been around for quite a while. It was original developed at MIT, and it started in the late '50s or early '60s. They formally defined it as a method for modeling and analyzing the holistic nature of complex problems as they evolve over time. There's a tool set associated with it that has increasingly improved over the years, so it's a pretty mature technology. They've used it since the early '90s, late '80s to analyze aspects of software problems, software process, but there's only a couple of instances where they've used it for analyzing software acquisitions. So that's where we are really bringing something new to the table here.

Jay: What are some of the aspects of the situation, such as an acquisition program, that you'd typically try to model?

Bill: Well, let me take a stab at that one, Jay. Really in simple terms, we can divide it up into two categories: One would be quantitative aspects, what we think of as hard data, the ordinary things you'd be monitoring [in] a large software development or acquisition program, the cost, the schedule, the staffing, perhaps information about defects, the other things that we very

commonly will measure about a given program. The fact is, while all of these things are clearly very important to be keeping track of, it's not the extent of what's actually going on in the program. There are a lot of other aspects that are a little bit more qualitative—what some people might call the "*soft*" information—that also can be very relevant to the way the program functions. These could be things like *confidence*. In other words, "Do the sponsors or the funders of the program have confidence that the program can actually execute as it claims it can?" Perhaps *satisfaction*, which is a little bit more difficult to measure, as well, of the receivers of the system; even things like the level of *cooperativeness* of the different players within the program. Those are the kinds of things that we're looking at trying to express in explicit terms in our system dynamics models.

Andy: Bill, another important point is that some of those qualitative concepts can really be important, but difficult to quantify. I know that kind of doesn't make sense in a way, but in a sense, if you're going to build a simulation model, you have to figure out how to quantify those things. The fact that they're difficult to quantify doesn't mean we shouldn't include them in the model. Estimating their influence as part of the model is important even if it's only approximation, and is surely better than not including those aspects at all, which would assume that their effect is negligible.

Jay: Okay, so what are some of the key insights that you've gotten from analyzing the preliminary model that you guys have worked on?

Andy: One of the key insights is that we discovered a *tipping point*. Typically, in a science project development, artifacts are transferred directly to a production development. What this means is that because the standards for a science project development are somewhat lower - the idea there is to test concepts - the quality is not as high. So, a lot of undiscovered rework is transferred from the science project development directly to the production development.

So, this undiscovered rework causes problems in the production development because the developers don't know about this undiscovered rework, and so they're not planning for it. And it causes problems later on. It creates a tipping point in which the production development cannot recover from this large amount of hidden rework. The developers get overly consumed by fixing the fundamental problems with the SP prototype (the science project prototype) in the form of architectural problems, and defects that are hidden within this system. Another issue is that this contributes to the "90 percent done phenomenon." I think Bill has a really good description of the 90 percent done phenomenon. I'll let him take that.

Bill: Well, sure. One of the things that we often see when we're looking at larger-scale software development efforts is that when the project first starts out, initially we're ticking off progress at a pretty regular basis: 10%, 20% done, 30, 40, 50, et cetera, but what can happen is as you start to approach nearing completion, the 70, 80, 90% done, is that progress as measured can begin to

stall out. It looks like we should be linearly progressing to 100%, but that just doesn't happen. There are a lot of explanations offered as to why that seems to be the case, but most managers of large-scale software development see this kind of thing on a regular basis. One of the outputs we've seen from our initial analysis of the model is exactly that and seems to display the same kind of behavior that managers are so familiar with.

Jay: Really interesting. So, the results and insights that you've found, how can they be leveraged?

Bill: Well, there are a number of different things that we can do. A lot of it has to do with—after we've done this analysis, and we've come to some conclusions about what's going on with the program's behavior—we can then begin to make some specific recommendations as to how these kinds of programs ought to be treated, especially in the future. One obvious one is that if there is going to be a transition from the science project into a larger-scale production effort that is going to involve reusing this prototype code from the science project, well, that's a bit problematic. Clearly, you want to be doing that—if that's your approach—as early as possible because otherwise the amount of rework that, as Andy pointed out, is undiscovered is simply growing. So, the longer you wait, the more of this undiscovered rework you're going to have, which is then going to frustrate both the developers and the managers in trying to turn this into a production quality system.

Another piece of guidance that comes out of our analysis is the fact that that may not be the right approach at all to take. In other words, you might be better off getting rid of the prototype early on, not trying to build on it all, and you're going to be getting overall, according to our model, a lot better program performance.

Jay: So, throw away the prototype, and start over is the best approach.

Bill: It is, but one of the things we talk about extensively in the technical report that we produced on this from the model results was to say that that has its own complexities too, because people—once they found out the value of this new prototype system—they want to start using it in the field as soon as they can. If what you're telling them is that, "Okay, we have to throw away this prototype now," which they're actually starting to use, they're saying, "Well, wait a second; we're finding this really useful in the field," in whatever application it might be. If the program has to "go dark" as they say, and not produce any results for a while, while they rebuild everything that the prototype was already doing, that doesn't make necessarily the people who want this so much, and want it so urgently in the field, very happy. So, this is part of the inherent dilemma that you find yourself in with this kind of dynamic.

Andy: There might be some intermediate line between throwing it away and completely starting over - if you understand the dynamic that is occurring here, and you understand that if you

consider it in an evolutionary frame of mind from the beginning - that might make an easier transition with less undiscovered rework later on.

Jay: Okay. So, how do you know if your model is correct?

Andy: Jay, this is a question we get all the time. It's a great question, but I'm going to tell you it's really the wrong question to ask. Any model is an abstraction of reality, and so it's incorrect in some aspects. The question is: "Is it helpful? Is it insightful?"

The way we've started viewing the model is as an evolving theory; some aspects of that are more grounded in data—data that you have. Some other aspects aren't, just because you don't have the data yet. But it embodies a set of hypotheses that you can continually try to test as your research moves forward. You can look for confirming evidence. Either the evidence you find confirms the existing theory, the model as you developed it. Or, perhaps, it refutes it, and you need to revise that model in a way that is more representative of the evidence that you've collected.

Bill: One thing I'll add to that is that there are some standard approaches that are well accepted in doing system dynamics modeling. They include getting together groups of what we call "subject matter experts" who have direct experience with whatever the situation or the dynamic that you're trying to model is; and getting those folks, interviewing them, getting their specific opinions on what relates to what other aspects, and specifically in what way.

Another way is to try to compare the actual output or the performance of the model with historical program data. So, you can look at programs which the model describes, and then look at their historical data, and see if what you're getting out of the model more or less corresponds with that general trajectory.

Andy: That's a good point, Bill.

Jay: Okay. So from a pragmatic point of view, what are some of the ways that this type of model could be used by the DoD?

Bill: Well, let me take a stab at that, Jay. You know, first of all is the way in which we're working with it right now, which is research, to try and get a better understanding of what's actually going on. We all can stand back, and we can see what happens, but we don't really have an understanding of why, and exactly how for that matter. So, here we're really trying to get, through our research, a much better understanding of what's going on there.

Once you have that kind of understanding—well, we've talked a little bit about it—your training is clearly one area. If people working on these programs were aware of the kind of trouble that they can be headed for depending on the type of program they're working, they can obviously take steps to mitigate that and reduce the adverse effects of it.

Another area is in policy development. Let's say that the DoD was interested in trying to figure out different ways of dealing with these prototype projects, these science projects, etc. They could look at the results from our model. We could even run specific scenarios, and they might decide that certain different policies might be more effective ways of handling those kinds of situations in the future. You could actually test some of those approaches with the model and see if you get a better outcome or not.

Jay: Okay. So, what is the future direction of your research?

Bill: Well, let me also take a quick cut at that, and Andy can chime in. What we're actually doing this year is we're working on a whole different model that is trying to look at the dynamic behavior of joint programs. So, joint programs are programs that are typically done between multiple services. So, you might have involvement by the Army, and the Navy, the Air Force, and the Marine Corps, etc. In fact, a lot of those programs, they're a great idea, because what we're trying to do is develop one system that all these disparate services can use despite the differences in the types of operations they run. Because you have all these different players involved—different groups want different capabilities in the system, etc.—so you can have a lot of challenges trying to reconcile those different views, the different requirements they have of the joint system.

In short there's a lot of complexity to them; both technical complexity in terms of what has to be built that will meet all these different demands, and organizational complexity of how do you resolve some of the different points of view and some of the possibly conflicting requirements that might come up in this. What's important, though, with joint programs is—because they help make better interoperability between the different services, because they help to reduce cost by building one system as opposed to different systems for each of the services, each one of which is custom—it's important to the Department of the Defense to make these programs as successful as possible. So, we're hoping that this new model that we're building now is going to help them do that.

Andy: One thing from the research prospective is, we are leveraging some of the existing research. There's not a lot that's been done on acquisition as I mentioned earlier, but there are some theories out there that we're trying to leverage and connect into. There are theories of cooperation and some actual system dynamics models in the area of negotiation that we think will be, and are proving useful in terms of understanding, documenting, and modeling those aspects between the interactions between different stakeholders. So, it's quite an exciting area for the research as well.

Jay: Really interesting discussion. Bill and Andy, thanks for joining us today. If you'd like more information about SEI's recent research you can download all of our technical reports and notes including Bill and Andy's report, The Evolution of a Science Project at

sei.cmu.edu/library/reportspapers.cfm. This podcast is also available on the SEI website at sei.cmu.edu/podcasts, and on Carnegie Mellon University's iTunesU site. As always, if you have any questions please don't hesitate to email us info@sei.cmu.edu. Thank you.