A Roadmap for Creating and Using Virtual Prototyping Software

featuring Richard Kendall and Douglass Post as Interviewed by Suzanne Miller

----

Welcome to the SEI Podcast Series, a production of the Carnegie Mellon University Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the U.S. Department of Defense. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

Suzanne Miller: Hello, my name is Suzanne Miller. I am a principal researcher here at the SEI. And today, I would like to welcome my colleagues, Dr. Douglass Post and Dr. Richard Kendall. Since 1975, Doug has initiated and led programs to develop and apply supercomputer modeling and simulation software for engineering design and scientific research at many places, including Princeton Plasma Physics [PPPL], Lawrence Livermore Labs, Los Alamos Labs, and the DoD [U.S. Department of Defense]. Dr. Richard Kendall is a software engineering consultant with the DoD High-Performance [Computing] Modernization Program, otherwise known as HPCMP. He has more than 50 years of experience in applications of computational mathematics to engineering problems.

Today we're here to talk with both of them about their book on creating software for virtual prototypes. The title of the book is Creating and Using Virtual Prototyping Software: Principles, and Practices by Douglass Post and Richard Kendall. The book is based on their extensive career experience and is especially informed by their experiences in CREATE [Computational Research and Engineering Acquisition Tools and Environments], a multiyear DoD program to develop and deploy software to create digital siblings for systems like ships, air vehicles, ground vehicles, and RF [radio-frequency] antennas, and to enable engineers and scientists to design these complex systems and to accurately predict their performance.

Welcome to you both. That's quite an introduction. I don't usually have that long of an introduction for people. You guys have been around a lot longer than some of the folks we talk to. Welcome. All right, so do you mind if I call you Richard and Doug? Is that all right?

Douglass Post: That's fine.

Richard Kendall: That's fine, yes.

Suzanne: Perfect. Let's start off by having you each tell us a little bit about yourselves and the work that you do that led to this book. So, Doug, let's go ahead and start with you, and then we'll follow with Richard.

Douglass: OK, well, I've had a career doing this [computational physics and engineering], starting with a Ph.D. in physics, that's experimental and theoretical physics at Stanford, and then have been using supercomputers, as you mentioned, at PPPL, and ITER, which is a large control-fusion experiment that's being built in Cadarache in France, and have led groups at Livermore and Los Alamos, and started a project at the Department of Defense [DoD], called CREATE [Computational Research and Engineering Acquisition Tools and Environments], and also have an affiliation with the SEI. Richard?

Richard: Well, my career also started right after a Ph.D. at Rice University in numerical analysis. And it's been dominated by the theme of applying computational mathematics to engineering. It also has had three phases, starting with an industry phase at Esso, now Exxon-Mobil. We didn't use the term *supercomputer* back then, and we wouldn't have called any computer we had access to a supercomputer. Exxon-Mobil was followed up with a phase in a startup. It was a venture to develop what we would now refer to as virtual prototyping software for the international oil industry. We did have supercomputers then: Cray 1, and CYBER 205. And then finally a government phase, first at Los Alamos National Lab in geosciences, and then IT, I was the first CIO there. Now with CREATE, for the last more than 16 years as a software engineering consultant. We're certainly in the supercomputer era now. We've just entered the exascale era. So, it's been a long ride.

Suzanne: Both of you have seen a lot of evolution in the computing field, not just in terms of the work that you're doing particularly with virtual prototyping, but software engineering methods, and our ability to actually apply computer science to these very complex problems. So, lots of places you could have gone different directions. What was the catalyst for the CREATE work, and how did you two begin to work together on this virtual prototyping software project?

Richard: Well, I'll start this. I think back in 2005 when CREATE started, the stars were aligned for it, so to speak. First, the Department of Defense desperately needed to find ways to speed up the development of major weapon systems. Second, there had been by that time exponential growth in the capability of supercomputers due to good old Moore's Law, kept doubling every year and a half or so. Doug recognized that virtual prototyping with supercomputers could meet this objective of speeding up the acquisition cycle. In particular, both of us had already met each other in the Department of Energy, and we had seen emerging successes in virtual prototyping there. We knew that this was feasible for multi-physics simulations that were going to be necessary.

We have a figure, and I recognize that those of you who are just listening can't see this figure, but I'll describe it a little bit. The first part of it describes the more or less traditional five steps in conventional product development, and the big bottleneck is in the middle of this figure. It describes the use of physical prototypes, both at small scale and large scale, to reach a product. This is a tremendous bottleneck, because for a sophisticated system like an F-35, it takes something like two decades to transit. That is, to get from the smallest scale to a full-scale product takes a couple of decades. That is just considered totally unacceptable to the department, so they wanted to do something about it. What we proposed was replacing that big bottleneck in the middle with virtual prototyping, where we replaced these physical prototypes with software and supercomputers and virtual prototypes to do the design and testing. It had the advantage of being dramatically faster and dramatically more comprehensive. We could do hundreds or even thousands of designs, test them, analyze them, not just a handful that might be possible with physical prototypes. So, this is the heart of what this is all about. Our challenge back then was to figure out how to actually create the software for virtual prototyping of the systems that the Department of Defense was interested in. We felt qualified to do that based on all our experience. Both of us have more than 50 years' experience. Our original proposal was accepted and has been funded now for the last 16 years.

Suzanne: All right. One of the things I want to say here is that the physical prototypes have always had the limitation of a fixed set of design parameters. One of the things that we appreciate about the virtual prototypes is their malleability, and once we understand the physics and have that model, that's the thousand-design possibility, where we actually have that malleability to be able to do that. So it really expands our trade space and allows us to make better decisions as engineers much, much earlier than when we're trying to run it in a physical space.

Richard: That's exactly right.

Suzanne: Doug, you want to add some into this and talk about your perspective on it?

Douglas: Yeah, for us the prospect of doing this was some of the capstones of our career in terms of that because it was a huge opportunity for us to apply what we had learned over 50 years of experience and apply it to a program. So the challenges were things like how do you get the proposal together, get it funded, and then how do you execute it? And so we did all of that, and we thought, well, now that we've learned these things, we have this CREATE program going, and new people coming in, and they're going to look and say, *How did it get started?*, and this sort of thing. So we wrote the book to document those lessons learned, and to make them available to the general community at large.

There were commercial examples for doing this. For instance, Goodyear Tire around 2000 was having trouble with their tire designs, they were losing market share, and so forth. So they decided to try this idea of virtual prototyping, and [they] went to Sandia National Lab and got some help,[Sandia] providing the coaching to do this and so forth. With virtual prototyping, they managed to turn things around. Instead of taking two or three years, they took less than one year to produce new tire models. So it just revolutionized their tire market, their ability to do tires.

Real challenges in this that we were beginning to face is when you start these things, how do you get stable funding, how do you recruit and retain the right kind of people to do this? How do you get the right staff? And how do you implement Agile methods for the government environment? We were finding that our experience in the DoE labs was really different than the experience we had in the Department of Defense. The Department of Defense is very rigid about things, with rigid requirements, rigid processes, and so forth. This is just totally incompatible with the kind of innovative development design, code-development practices that were actually working with real people who have to build real codes, because it's more like an invention process.

Suzanne: I want to point out just for a second, Doug, that 2005 was only five years from the Agile Manifesto. So, not only were you fighting DoD's typical expectations, but the methods themselves were, by many people, not considered to be proven yet, so you had that in addition. So yeah, you were in a very challenging time for making these kinds of changes on a very large-scale and important program.

Douglass: But what we did have as an advantage through our whole careers, was that these were actually the kinds of methods that were being used by people who were actually getting a job done.

Suzanne: Yes.

Douglass: Now that, of course, didn't matter to the Department of Defense. [Of necessity, it had a fairly rigid set of rules.] The Department of Defense is a large organization, 3 million people, billions of dollars — $700 billion budget every year. We had to work within that system and figure out a way to actually do Agile software development, and we succeeded.

Richard: Let me chime in and say that, like Doug said, DoD acquisition programs were expected to be like the hardware programs. That is, they [DoD] had a hardware mentality, and they expected software to conform to it. For example, planning was always long-term, milestone-plan based, tracked by the earned-value management method. Well, we recognized that that just wasn't going to work for us. We needed to be able to invent things and to be innovative. And you can't get there by a long-term, long-range plan. But we also recognized that our sponsors in the acquisition community would really be skeptical of us if we couldn't deliver. So we had to concentrate on trying to figure out how to be able to deliver. We had other challenges that we describe in the book. I'll mention one in particular and that is we found ourselves working with DoD R&D organizations, which were not accustomed to creating what we would call "production-level" software. That is, software that could be used by somebody who didn't write it. They weren't accustomed to doing that, and that made software engineering, in particular, a much greater challenge than it might have been otherwise.

Suzanne: You've already said a couple of things about some of the insights that led you to Agile software development as a foundation. In particular, I've heard you say that you're driven by innovation in this whole program. And you can't schedule innovation on a three-year plan, and *on month 9 I'm going to invent this, and on month 18 I'm going to invent this other thing*, that just doesn't work. Agile, we know, is very amenable to that kind of thing. What are some of the other things that led you to Agile software as the foundational method for how you wanted to do work? I'll start with Doug.

Douglass: As we were mentioning, it's a creative process. And it's also one of the other standard problems that the regular Department of Defense program has is setting requirements. For the case we were having to do, you can't really set requirements for things, because software development is a requirements-discovery process. You don't know what you're going to get, and so really what you end up having to do is to have senior management define the goals for where you want to get to, and then select them, and then push the tactical decisions of what to do down to the lowest level possible—to the people who are developing the codes—because they're going to be the people who know more about what they're doing, and more about what the problems are, and are better suited to do the discovery process. You just can't develop software, as we were saying, with a top-down process. Another thing is that the time scale for these sorts of things, these physical systems, have very long lifetimes. The development of an aircraft carrier, it takes 10 or 15 years to design it, and another 10 or 15 years to build it. So you just can't turn those things around quickly, and so you've got to really treat it and match them to the scale of a task you're intending to do.

Another thing is that the Department of Defense had this idea, based on the hardware, that you put together a design, and you build it, and then you're done. Okay, maybe there's a little maintenance, maybe there's a little revision later on, but, you know, it's done. And so what we found with dealing with the Department of Defense was they kept asking, *When is the software project going to be through?* The real thing is it's never done in the sense that if the piece of software is being used by a user to do a job, it's going to have to be constantly refreshed and upgraded and so forth to adapt. If requirements are going to change, one of the great things about software is you can change the software to meet new requirements, and so it's never really finished. This, of course, has been a major issue, as the Department of Defense takes on more and more software. The Defense Innovation Board and the Defense Science Board have really stressed the idea that software is immortal. It lives as long as you need it, and if you stop supporting it, it dies, because it rapidly becomes out of date, the computer systems change, and then it doesn't work anymore.

Suzanne: Those are definitely things that we are living through, not just CREATE, but other systems. Richard, did you have anything you wanted to add into that?

Richard: Yeah, let me add that one of the other things that we were really focused on in CREATE from the beginning was the management of risk. Software development is an extremely risky undertaking. Failure rates in the literature cite it at 60— 70 percent. In particular, a recent *Information Age* article described a failure rate of 71 percent, the upper end of what I just quoted. Doug and I have done case studies in the past in which we discovered that software very similar to what we're talking about in CREATE had a failure rate of 50 percent. It was a very sobering case study, and it motivated us from the very beginning to try to manage risk. There have been billion-dollar government failures in software development. The FBI Sentinel case-file management program was a billion dollar failure, and so was the NSA Trailblazer software program. When I say failure, I mean no working software after that investment.

Suzanne: Yeah, if you've been around the DoD a while, you've seen less impactful from a dollar viewpoint, but still very impactful failures. The management of software risk is one of the things that seems to be a common failure mode in a lot of those, or the lack of management. So you have all these experiences, you have been very successful in using these new methods. But being involved in adoption management for Agile and other kinds of practices, I know that this was not an easy path for you. So, talk for a minute in terms of things you talk about in the book related to the effort that you needed to implement this switch. Not just for the… R&D folks that aren't used to building products, or the DoD senior leaders who aren't used to hearing things like, *I cannot give you a three-year schedule with defined dates*. Lots of people, lots of stakeholders had to change the way they thought about and behaved in relationship to your project. So, tell us a little bit about that experience. Doug, do you want to take that one?

Douglass: I'll take that. The most important thing is to really recognize how hard change is. OK, so you go and read a bunch of books about change. There are a lot of them out there. Machiavelli is famous for pointing out that there's nothing as hard as change. You're going in and upsetting some standard way of doing things, and everybody's going to fight you because of it. There's really no incentive, and no one else has any incentive to do anything, because it's a big risk, and they don't really benefit very much from it. So it's really very hard. What you have to do is to really work hard to explain to people, get them to understand why they have to change, and what the advantages of all this. You have to engender a certain amount, a reasonable amount of trust in what you're doing by management, and it's really scary. So, in most cases in the literature and so forth, change is easiest when an organization is facing an existential threat. In other words, if you don't change [your] business—the way you're doing things —you're going to go out of business.

It happened with Goodyear Tire. If they didn't change their tire-development process, they were gone. They were going to go bankrupt. They recognized that, and so they were looking desperately for something to do. A fellow named Loren Miller at Goodyear figured out that computational modeling was a way to try it. Because they were facing this existential threat, he got a chance to do this. He did it, and it's paid off very well for them.

The DoD faced similar challenges as Goodyear.   Up to 1975, it took about five years for a new airplane to come out. The same was true of cars and so forth. But around 1975, military airplanes got more and more complex, and it turns out the design and production times went from 5 years to 10 years to 25 years over the next 30 years. If it takes 25 years to get a new airplane put together –F-35 was an example, the Osprey was an example—that's too long. I mean, you pull out Augustine's book where he points out that in 50 years, at the rate at which the expense of airplanes is growing, 50 years from now, the Defense Department will be able to afford only one new airplane every year. That single airplane will be shared between the Air Force, the Navy and the Marines and the other organizations, and that's just unacceptable.

So that's really what convinced people that you need a better way of doing things. That's become easier as we get into this. There have been a lot of quantitative studies done now that are now beginning to be published. This stuff was all very sensitive for a while, because nobody wanted to own up to the fact that they just weren't doing a good job, that acquisition process wasn't working. And now people are much more aware, and it's become more obvious, and things that are improving. There's been a DARPA study [Competing in Time: Ensuring Capability Advantage and Mission Success through Adaptable Resource Allocation] published by the Hudson Institute fairly recently in February, 2021 that is described in our book, but you can also get it off the Hudson webpage [https://www.hudson.org/research/16717-competing-in-time-ensuring-capability-advantage-and-mission-success-through-adaptable-resource-allocation], that points out that this situation was really dire, but it's now getting much better.

Suzanne: Yeah, and I would argue that some of that sensitivity was because in 2005, we didn't have confidence in solutions to these problems. And one of the great contributions of CREATE has been to prove that we do have solutions to these problems. Once we have solutions, then we…it's human nature… once we have a solution, we're actually more willing to talk about the problem, I've observed.

Speaking of the success of CREATE, what is its current status? And to what extent has it been put into use by the DoD? That was the ultimate point is to improve the acquisition, both the quality of the products we're building and the acquisition cycle. So, Richard, why don't you start us talking about that?

Richard: OK, first, CREATE has touched a lot of programs. It's touched at least 40 DoD programs of record with an aggregated budget of something like $50 billion. It has 2,500 users and customers, in government, industry, and in universities. Now, 2,500 may not sound like a lot if you're Microsoft. But in our business, 2,500 users of a product like ours is a lot of users. It's almost everybody in the Department of Defense who could possibly use it. It's a lot of users. We're at our 16th year now. I think we're firmly established as a part of the Department of Defense's migration toward digital engineering. Sixteen years, by the way, is a long time for a software program in the Department of Defense. They're typically three years long, and we've managed to develop and mature, and I want to emphasize the word *mature*, 12 major systems, four for air vehicles, one of which is shared with the Navy for the design of ships, five for naval combatants, one for radio frequency antennas, one for ground vehicles, and a gridding and geometry tool that supports the whole CREATE family. These are not just in their infancy. These are quite mature now. They have more capabilities than almost anybody, any one person, could ever understand.

Suzanne: That is very successful in our experience with DoD software programs. And the breadth of things that you've been able to do, and I want to emphasize, these are not DoD business systems. These are complex cyber-physical systems. These are hardware/software systems, so you're really attacking the problem of the really expensive and long-lead kinds of items. That is something to be really proud of, Doug, I know you have some examples of CREATE. Doug, I know this excites you, so let's talk about some other examples.

Douglass: It's really nice too, because you work with people who are very excited about what they're doing. So you're working with the top-level engineers and scientists at the Navy and the Air Force, and in the Army aviation program. To be able to work with people who are putting together a piece of software that can calculate how a helicopter flies. Helicopters are amazingly complex, and we can actually predict, those guys can predict how all this does.

Examples of, I'll just quote four or five examples of tools. The Navy has got a new set of submarines coming out, both a ballistic missile launch, and for attack submarines. A new generation is coming out; and our tools are being used to design those systems, to measure their performance, to predict their maneuverability and sea-keeping properties. We are designing tools for the new helicopters that the Army aviation people are putting together. These tools are being applied to new fighter airplanes, like the F-35 and the upgrades of the F-15 and so forth. They're even being used to design the parachutes that are guiding the dropping of cargo into active theaters.

Suzanne: Wow.

Douglass: You wanted to go in and do this—you want to design them so that they don't hang around, the airplane doesn't have to hang around trying to wait to find the right place, while people are shooting at you. And another example is they're upgrading the B-52, putting new engines on it and changing aerodynamics somewhat, and the tools are being used to keep the A-10 alive, which is the plane that provides ground support for the Army. So those are just examples. They're illustrated in the picture that will be available in the video.

Suzanne: Yes. And so for our audio listeners, we do have video that we're shooting as well, and the images that Richard and Doug are talking about will be available through the video.

Let's talk a little more about the audience for this book. Some audiences are going to be interested in what CREATE has done and how you manage to make these very complex computational models. But you also mention writing the book for people who follow you. Both of you are well deserved of getting into retirement and doing more fun things. So, for those that are taking CREATE forward and things like it, that's one audience. But who else do you think would be interested in this book? Richard, why don't you start us off?

Richard: Well, of course I think all along we intended it for engineers and scientists, a broad category of engineers and scientists who want to take advantage of the power of virtual prototypes to improve their competitiveness. But as we were inspired by the Goodyear story—they were trying to make themselves more competitive with this—our hope is that it will help the readers plan and manage and shift to virtual prototyping, establish and execute Agile processes, manage the impact on development workflows. This really impacts a workflow. It really changes a workflow. Verifying and validating the software —there are several chapters in the book that touch on that activity. Recruit and retain the specialized workforce needed to do this. This isn't routine software development. And look for new opportunities. These are all things we hope our book will help readers get some insight into.

Suzanne: I especially encourage readers who may not have seen themselves, Agile methods, Lean methods being used in very complex engineering software, people have a tendency to say, *Oh, yeah, you can use Agile over on the business side, but when we're dealing with these complex cyber-physical systems, we have to go back to traditional systems engineering and waterfall methods and things.* I found that this is very inspirational in terms of saying, *No, you don't, you have to change your workflow, you've got to change some things, you have to work differently together. But it is absolutely as applicable here as it is anywhere else.* And I think you guys made a great case for that in the book. I have people I'm going to be sending chapters to just for that purpose. All right, so we are talking about the Agile and some of the things about verifying and validating software in this complex kind of interface. One of the things that I think is notable about the book is that you're looking at this process-bound environment, like the DoD, but you're talking about some of the issues that you don't learn about in DAU [Defense Acquisition University] and you don't learn about in engineering schools. And I think that's another…how to put a proposal together, get decision-makers to pay attention to it. Those topics, they are not typical in textbooks. So I'm curious as to what made you decide to write about those things. Richard, I think you wanted to say something about that.

Richard: Well, you brought up one point already, and that is when we started this, there was no example game plan within our community of interest, of engineers and scientists, for how to turn an opportunity for virtual prototyping into a successful program. We had nowhere to turn to see that. We weren't aware of the rather extensive literature in Silicon Valley for this. It's just not a part of what we were exposed to, as you pointed out. We weren't taught to do this. And so we felt like we needed to address that need. Chapters 6 and 7 of the book, [describe] our game plan for doing this, in case you want to try it. And then, of course, the other big, big obstacle for us was the fact that the then-existing program-management, software engineering literature didn't provide any guidance whatsoever into how to cope with the hardware focus—I really call it a bias—of DoD Instruction 5000, which is the document that governs the DoD acquisition world.

Suzanne: And you were under what was called 5000.2, which is hardware-centric enough that they actually made…5000.87 now exists, since 2019. It doesn't address software, but you didn't have that.

Richard: No, no, we didn't have that. In fact, if you look back at the material available to us in 2005, there was really no distinction between software and hardware.

Suzanne: Yeah, you're right. I remember the 2008 version started to kind of pull apart, but you didn't even have that one yet.

Richard: Right.

Suzanne: So, you were embodying a lot of lessons learned in this book. We've talked about lessons related to how do you get this thing going, how do you make decisions about it, how do you get change among the stakeholders. Doug, do you want to give us a few other lessons learned that you want to add into this list?

Douglass: Let's see, the book really addresses…what we felt was unique about it. It really addresses the practical issues, not…you know, not what everybody in the business has gone to engineering or physics school and graduate school for. They know how to do the technical details. But they've never really learned how to put a proposal together, and how to deal with complex organizations, and how to get funding, and the importance of putting all of this together in a real organization.

I mean, one of the things that I learned fairly quickly when I got there was from John Bramer, who was the chief of staff at the SEI. He gave me a book by Hugh Montgomery called Bureaucratic Nirvana: Life in the Center of the Box, how to survive in the box (the Pentagon). It was a really very nice thing, because he described it as kind of a cultural anthropology situation. You had all these people in the Pentagon trying to maximize their own effectiveness and their own careers. They're all well-meaning, and all trying to do a nice job. None of them were really out to get anybody, but their competition didn't produce an optimal result for the organization.

But, you know, there were simple things in the book, like if you want to advance your career, the way to do that is to start a new program. Well, to start a new program, you need money. Well, there is no new money really, and so you've got to take money from somebody else. So, your job devolves from running a program and getting it established, your job is to protect the money. If you can't do that, you don't have a program. So, I guess [if] you don't have any money, you don't have a program, and so it's learning all these practical things. It leads you into things like figuring out how to get a proposal done, figuring out how to work with the financial people, because the financial people in any organization are the people who really control, you know. The management trust them to make sure that funds available to the organization are spent to get the job done. If you haven't got them on your side, it's a hard road. So you've got to bring them in because normally they're people who scientists and engineers don't really resonate with because it's not their thing. But they become some of the most important people in your professional life, because you need them to get aboard, and you need to get them to understand what you're doing, and the value of it.

Suzanne: …and to be able to communicate it. That's the thing I value.

Douglass: Value to their management, but we're relying on them to earn that trust, to get that trust. And so that's an extremely important thing. I learned that fairly early, thanks to John, and his advice on other things. Probably one of the most useful things that we did in putting CREATE together was getting the financial people in our organization, to understand this and trust us. Then as life got on, it got more and more bureaucratic in the funding business, because the government was looking harder and harder to do this. Having them on our side was just a wonderful thing, and that was really a win.

You've also got to put an emphasis on establishing the value of software applications and recruiting and maintaining software developers. That turns out to be an incredibly important issue. These people are precious beyond belief. There are very few people who know enough about computing and aeronautical engineering to design an airplane. And if you've got people that are really able to do that and do it well, then you really want to protect them and treat them very well. They're really the most valuable people in your organization. Everybody's valuable, of course, including [support staff and administrative staff]. But a really talented, good aeronautical design engineer with computational expertise is necessary. If an airplane is not well designed, it's going to fall out of the sky and that's not the right answer. And so, and then finally, as we mentioned, verification, validation of the software is something you can never lose sight of. You have to keep working on that. Because if you get the wrong answer, again, that's not the right, that's not going to be successful.

Suzanne: I also remember reading you were very adamant about confronting risk and not letting risks fester and not letting them sit. Say a little bit about that. Because I think that's, risks are difficult because whoever raises the risk is kind of, you get into a shoot-the-messenger kind of position. But if you don't deal with risks, then they manifest. Say a little more about how you manage that.

Richard: Let me try. Well, we were shocked, as I said earlier, when we did this study before CREATE started, and found that half of the software programs that we studied were failures. We had read the literature about this, so we knew that failure was talked about a lot. But to see it in things we understood was shocking. So we sat down at the beginning of CREATE, literally at the beginning, we asked ourselves, *Where are our biggest risks*? I don't know of any other program I've ever been involved in in all these years where we did that as the first thing. We asked ourselves, *Where are we most likely to fail?* Because we expected CREATE to live a long time, it wouldn't be successful if it was not a long-term project. So we asked, *What is going to prevent that from happening*? And we tried to identify the worst risks. Of course we identified dozens, but we tried to focus on the top 10 risks. There are things that are obvious in some ways, but people don't really focus on them and identify them at the beginning. Long-term funding, that's an obvious risk. But if you're not constantly paying attention to it, you will experience it, and you'll fail, and there are many others. I don't want to try to recite them all from memory. Doug has mentioned another one, the very important role that developers play in the process, and how to keep them engaged. That's a big risk.

Within the department[DoD], sponsor support is a big risk. Our sponsors are people who change jobs every two or three years, and we might have a new sponsor who is not at all sympathetic with anything we're trying to do. In fact, [someone who] wants to do exactly what Doug described, which is to capture our funding and apply it someplace else. So, these are examples of the kinds of risks that you have to pay attention to. Actually, technical risk is [important], I think— Doug can chime in on this— but it's further down the list. The ones that you have to worry about most are really not technical risks.

We felt, because of our experience in the Department of Energy, that there was a handle on the technical risks. We had seen them addressed, and we thought we could do this. We didn't <u>know</u> we could do it, but we thought we could. But we didn't know whether we could manage these other kinds of risks, like I've just described. So, from the beginning, we identified them and began to try to address them and track them and mitigate them, and some of them are extremely hard to mitigate. We can't mitigate the promotion profile of managers in the Department of Defense. We can't really mitigate that. But we can try to work with the process and make sure that the people who are coming along know who we are. So, that's an example.

Suzanne: Good examples, and going back, I think the theme of this whole conversation is all of you good engineers and scientists out there, you're not going to be able to meet your goals unless you deal with these programmatic and sociotechnical risks is what I call them, the ecosystem that you live in. If you ignore that, then you're not going to be able to get your goals met. So, even though it may not be your favorite thing to do, you're going to have to deal with it, or find somebody to partner with to deal with it. I definitely appreciate that perspective.

I want to switch over to a theme for our podcast, which is transition, in the sense of, for people that are interested in learning more about the CREATE approach and the CREATE products, how should they get started? What kinds of resources are available in relationship to learning more about CREATE? Doug, of course, and Richard, the book is kind of at the top of the list. But beyond the book, what are some of the things that they can look for? And I guess Doug or Richard, whoever wants to take that.

Douglass: I'll speak to one thing. Doing these sorts of things, developers of course, are your real assets. But they're not going to want to do everything you want them to do. And one of the biggest resistances we found, or at least I found, was getting them to do documentation. They're writing the software, this big, huge complex thing, zillions of statements, huge programs, multiphysics, complicated mathematics, complicated physics, complicated computer programming, and so forth.

There's going to be turnover in the computer business. I would say we're lucky to have people in the same job for five years. Yet the program is supposed to last 15 or 20 years. And so how do you, the new person, the new employee, come in, and look at this extraordinarily complicated airplane-design program? What kind of document are they going to have to pick it up. The person who wrote it may even not be there, and certainly is not interested in holding the hand of somebody for a couple of years until they've learned this thing. So you really have to get people to do the right kind of documentation and pull it together, and they're not going to want to do it. I remember that was one of the things that the groups were really very reluctant to take on until you convinced them that it was really essential, and they had to document the code because it was in their best interest. So they had to put together a user manual, because they wanted the code to be used, so they had to make it easy for people to use. They had to train users and things like that. If they didn't do that, then they were going to be on the phone all the time trying to tell somebody exactly how to use the code.

You have to put together a test plan and let the users have the test plan and the test results so they could get the practice of getting the right answer. Because getting the right answer is not just having code that has been verified and validated. You have to validate and verify that the users can get the right answer. So all of these things are really important. And documentation is something you need because you're not going to remember what you did five years ago either.

Suzanne: I don't remember what I did five years ago.

Douglass: And especially in writing complex code. So it's got to all be there in a way that people can actually make sense of it and use it.

Richard: Well, let me chime in. We have a slide. Of course, it won't be available for you if you're just listening here. But the CREATE developers and management have published a lot. There are 16 papers in the IEEE Computing in Science and Engineering journal that you can consult for various aspects of CREATE. The various product-development teams have been prolific publishers. For example, the air-vehicles team has probably published something approaching 200 papers in the AIAA [American Institute of Aeronautics and Astronautics] journals. The ships team, the team that develops software for naval applications, has published extensively in the American Society of Naval Engineers journal. The other teams have also published in various journals. As far as CREATE itself is concerned, it maintains a public-facing website that has a lot of information about CREATE in particular, and about the kinds of applications of this sort of software, virtual prototyping software. You can find it by simply searching centers.hpc.mil for CREATE. So, there are lots of publicly accessible sources for information about not only the CREATE program, but the general concept of virtual prototyping with many, many very specific examples of how the paradigm can be used in the literature.

Suzanne: Lots of resources. A lot of them self-serve, and so that's important. And over time I imagine, I would expect to see some of your work, especially now that you've got essentially the textbook, become part of the curriculum at graduate level, I would hope, so that people actually learn more of these things before they get out of school. But we don't control that, so we'll see how that goes.

What is next for both you, Richard, Doug, and the work? Are you anticipating another book? Do you have other plans that are related to being involved in CREATE? Or is it, are you going to get off of this stage and go do other things? Doug, I'll start with you and then Richard.

Douglass: I'm not sure we're going to write another book. It's a lot of work. Probably have to do a few more blogs and podcasts.

Suzanne: OK.

Douglass: To go into these things we have expanded on, for instance, the stuff [relating to] DoD, expand on the kind of documentation that's useful, expand on actually probably the most important thing—on how you deal with the human dynamics of having a lot of very bright people working with you and for you, and how you treat them, and how you make them feel, how you help them grow their careers, how important it is to help their careers grow professionally, and what kind of impact that gives your program, and helps you recruit new people and so forth. Those kinds of things we think would be nice to do in terms of podcasts and blogs. But I don't, I've been looking through the book recently to prepare for this and other things, It would be nice, we would write a somewhat different book. But it's an awful lot of work.

Suzanne: I know.

Douglass: We worked on this for almost five years. It's a lot of work. The publication process is too.

Richard: Yes, it is.

Suzanne: Nobody's going to fault you if you don't want to take that on again.

Douglass: Both of us have had 50-year careers, and now I'm 77, and Richard's almost 80, so yes, we're retired.

Suzanne: You might want to do some other things. Richard, what about you?

Richard: Well, yes, actually, I'm not retired. I'm still working with CREATE, and I hope I can continue it. I'm their software engineering mother confessor.

Suzanne: OK.

Richard: I'm still engaged with the program and hope to be in the future. You know, it's 16 years old, it's very mature. But it's clear that it has a lot of potential, a lot of potential. We've had many, many successful applications. But it's just the beginning. We're just at the beginning of this. Doug and I were involved in programs in the Department of Energy, in software development programs that were 50 years old. Started in the 60s and still active.

Douglass: 50s.

Richard: Yes, 50s even and still active. And so we hope for something like that for CREATE and other things in the Department of Defense.

Suzanne: Well, I think with the two of you shepherding this kind of content forward, I think that you're going to still have some fun with this. I think, when I get into my 70s, that's going to be kind of my goal, is whatever I'm doing, it's got to be something that I want to have some fun with. And I see the passion in both of you for this. So, I think the fun is still there. And as long as it is, we're going to get some great things out of you guys.

Thank you so much for giving us this time. I do want to make a final note. We would like to thank the CMU Software Engineering Institute (SEI) and the DoD High Performance Computing Modernization Program for supporting the writing of this book. It is available from Addison-Wesley at informit.com, and on Amazon. That will be in the transcript.

I am so pleased to have gotten a chance to talk with you. I don't know if you even remember me helping to facilitate a workshop for you back in 2008. I remember some of those early days and some of the things you were dealing with. I can't tell you how pleased I am at the success that CREATE has had, and your ability to translate that into some really valuable lessons learned for our community. I look forward to doing some more podcasts with you in the future.

I do want to say that in addition to other published works, which we will link to, Doug has also authored a post about this work on the SEI Blog. So, you can go to our blog site, insights.sei.cmu.edu, search under Doug Post, and you should be able to find that. I do want our listeners to know that we thank you for joining us. If you can get to the video, you'll get some extra goodies from the visuals, and we will include links in our transcripts to all the resources that were mentioned. I think we might even be able to dig up your Machiavelli reference, Doug. I know where that one comes from.

I'm thrilled that we were able to have this time together. I look forward to seeing what new ideas you come up with and new successes for CREATE and its applications. So, with that, I will say thank you all and have a good rest of your day.

Editor's Note*: The authors would like to thank the SEI for their support of this work and their support of CREATE from its inception. This transcript has been edited for clarity.*