Oral History of Barry Boehm, part 2 of 2

Interviewed by:
David C. Brock
Lee Osterweil

Recorded February 20, 2018
Los Angeles, CA

CHM Reference number: X8388.2018

David C. Brock: Well, great. Thanks for joining us again, Barry.

Barry Boehm: Sure. This is great.

Brock: If my memory serves, we were just at the cusp of talking about your involvement with—going into your involvement with DARPA when we broke from our last conversation.

Boehm: Right, yes.

Brock: I thought maybe we could just back up a little bit and take a little bit of a running start to get back into that. If you could tell us about—take us back to the kind of work that you were doing with TRW in the late 1980s, and your association with DARPA [Defense Advanced Research Projects Agency] over the years, and how this opportunity to work with DARPA more directly came about.

Boehm: Okay. That sounds good.

Brock: Okay. <laughs>

Boehm: At TRW I was involved in two generations of building a corporate software development environment. The first one was about the first time that we had local area networks, and so we were getting people in networks together. We actually used the COCOMO [Constructive Cost Model] model to estimate whether we would be able to improve productivity. The company is a conglomerate whose biggest thing outside of defense is auto parts, and they were sort of psyched by the Japanese Fifth Generation initiative and said, "Everybody in TRW has got to get more productivity. And if you have a good proposal for it we have corporate funds to do things with it."

Basically, we came up with a plan for this and I used the COCOMO model to say, "If your tools are better than this, if you educated your peoples in these technologies, you ought to be able to double productivity in 10 years." We put that forward, and my division general manager thought this was a really good idea, but he said, "We've got to hook you up with an upcoming project so that we know that what you're doing is going to fit what they need."

That turned out to be really good, because in a lot of cases I hired a couple of people: Art [Arthur] Pyster, who was the head at UC Santa Barbara, and Lolo [Maria] Penedo, who had just graduated with a PhD from UCLA. So we had a very strong technical group of people, and they had a lot of good ideas for automated test case generation and things like that. And the project people said, "What we really like is any time we make a change in the document to put a little bar at the right that says, 'This thing has been changed.'"

<laughter>

Boehm: This re-baselined everybody as far as what the real projects priorities were going to be and things like that. At any rate we built one and it really did double productivity for that the project and then did for a bunch of the future ones. Looking at things, we found that of most of the time people were producing documents and filling out forms rather than writing computer programs. So, we made sure that the secretaries would get on this. This improved productivity a whole lot because the software people didn't have to retype things and the like.

We looked around for technology in forms management and found that IBM and DEC [Digital Equipment Corporation] had some things, that they were pretty clunky. We felt we could build a better format forms management system than they had. So we did—and we ended up with the world's best 1982 Forms Management System. By 1986, we still had the best 1982 forms management system <laughs> and because we were amortizing its maintenance over one company, and the computer companies were amortizing it over lots of things, they got a lot better. We did a second generation where we were doing much more commercial technology involved in it.

Around that time, DARPA was getting interested in software engineering environments, and there was this project called Arcadia that Lee [Osterweil] was involved in, and Dick [Richard] Taylor and a number of people from UC Irvine were involved in. We became sort of their industrial counterpart and saying, you know, "Here are some things that would be valuable to make sure that the industry people would be interested in them," and making some contributions. Like Lolo Peneda came up with the project master database and that sort of thing. DARPA got quite interested in Arcadia, and there were various other things that I was doing that connected me with DARPA.

At that time the director of the DARPA Information Science and Technology Office was a really outstanding mathematician named Jacob Schwartz. Unfortunately, he was somebody who wanted to put his hands on everything that was going on in the office and sort of micromanage the program managers. He basically had a disinclination to do things that that weren't mathematically oriented. He wanted to kill the artificial intelligence program.

When I got there, there was a proposal and a draft budget to do that, but I got some of the AI people to come and say, "Should DARPA really do this? What kind of things, if we keep it going, do you think are going to make a difference?" We got the Marvin Minskys and then the Ed [Edward] Feigenbaums and the like together and they made a pretty strong case, and so I was able to sell the DARPA director in putting the money back into AI, which probably, in the first month I was there, the most important thing I ever did.

<laughter>

Brock: Yes.

Boehm: Since it just sort of took off like a rocket a few years later.

Brock: May I ask you to speak about the context in which DARPA would become interested in… You know, why was there the interest in the Arcadia program and for these software engineering environments? What was the larger frame for that concern?

Boehm: There were two—

Lee Osterweil: So, sure, yes, I can help with that.

Boehm: Yes.

Osterweil: Because that was our project. At the time, back in the '90s, people were building all these tools. There was this proliferation of software tools, and everybody had a tool that did something slightly different. Our rationale was just simply, you know, a carpenter has a whole toolkit and which tool to use at what point in order to build a house. And a software engineer has all these tools lying around all over the place, and they all do something different, and, you know, you ought to have a nice box for them. The tools all belong to the right place and people know what tool to use at what time and in what way. We called it an environment.

Other people had used that term. But we sold the idea of an environment on diverse tools where the DARPA angle was it wasn't the tools that we were building, it was the box. It was the holder, it was the framework for integrating the tools. DARPA bought off on that, and we had a pretty good run of six, eight years. We never produced something that became something that industry picked up and sold, but it was a blueprint for a lot of things that have come since. I think DARPA liked the idea that they had built a framework within which you can build more interoperable tools, although Barry may disagree about what DARPA saw and that was what we thought we were selling.

Boehm: That's basically it. And I think really, doing a lot more in terms of, you know, what's the architecture of one those things going to be like, and how do you do traceability among the design tools, and the requirements tools, and the code tools, and the test tools and things like that. That was one basic thing.

The other was that there were some other initiatives in software within DOD [Department of Defense]. One was called STARS, for Software Technology for Reliable—Affordable Reliable Software, or Systems.[1] The Defense Department had decided that it needed a better programming language for doing defense software. And the Ada initiative came along. The STARS program sort of evolved into something that was going to be a set of environments for programming in Ada. The DOD needed somewhere to put that, and then they put it at DARPA.

Brock: I see.

[1] Software Technology for Adaptable, Reliable Systems. Added by Frances Corry on August 31, 2018.

Boehm: I had a couple of program managers, Jack Kramer and John Foreman, were sort of running STARS at various times. What we ended up doing was getting bids from companies that would develop three different environments: one oriented toward the Army, one for the Navy, and one for the Air Force, and getting people from the Army, Navy and Air Force to be sort of the customers for this, and also saying that the companies needed to coordinate this with the other software environment things that they were doing. IBM and Univac basically had their own non-Ada environments. We were trying to get them to at least have something where the Ada environments and their other environments were reasonably interoperable, and maybe used the same requirements languages but went down to different programming languages when you went along.

This was a major initiative. The three winners were really IBM, which had its software environment in, actually in Toronto, and Univac had its in Minneapolis. The other winner was Boeing, who basically had done a strategic partnership with Digital Equipment, DEC. So we had a major management challenge of trying to make sure that these people were doing something that was good for them, but also good for the Defense Department. We had the representatives of the services saying, "All that's very nice but we also need something that addresses some other kinds of propriety or security," or other kinds of concerns that they hadn't been concerned about.

We had monthly meetings, and at every monthly meeting we would try to come up with, what are our plans for the next month, and everybody would agree those plans are going to be compatible across the three environments. We'd all go off and do our thing, and the next month people would come back and they'd say, "You know, we went to the home office and we got to back off with this."

One of the things that I found out was an important thing that I hadn't seen any tool support for previously, which was if you have a bunch of multiple stakeholders who have conflicting priorities and things like that, how do you get them to negotiate and converge on and evolve the requirements and the boundary conditions and the maintenance models and things like that for these.

This is sort of leading to a jump into the future, that when I left DARPA I wanted to do some significant research and one of them was doing this kind negotiations system for requirements and support technology and the like. The underlying principal on it was that you had to come up with a win-win for all of the stakeholders. So, when I got up to USC [University of Southern California], we started building a WinWin system and after 18 months I could bid for DARPA dollars again because I was avoiding conflicts of interest before that.

Brock: Right.

Boehm: We did a bunch of generations of this WinWin system based on really pretty primitive technology when it started, and we're now on the sixth generation which looks more like Facebook. That's sort of a side effect of some of the things I was doing at DARPA.

Brock: Would you say that has been the main thrust of your research work after DARPA?

Boehm: It was in the first few years in, after I got to USC that was… Fundamentally, we teach software by having the masters students get together in teams of six people and get connected to a USC neighborhood client who wanted some software built for them. Every year we would have about 20 teams of these, and we would use the WinWin system of the day to address what was going on.

We had the various lessons that we learned there. One was we were trying do groupware on support systems that were oriented toward individuals. Basically, the earlier versions of WinWin basically said, "Here's a screen. Put first win condition in, post it to the database of win conditions and put your next one in, put your next one in." Everybody was thinking about themselves as they were doing this.

Later on, we got connected with a company called GroupSystem.com, which had developed a really good set of collaboration tools. When we worked with them on it they said, "Anytime somebody puts a win condition in, the next thing they should see on the screen is the related win conditions that other people are seeing." Somebody said, "We want this thing to run on all four platforms—that's Unix, Linux, Windows and then Apple." People would say, "Well, do you have copies of those that we can use to test whether we can do this or not?"

So there were a bunch of things that as people put something in and said, "Well, if you're going to do that how are we going to do this?" It really did converge things, and in a way that people would also sort of horse trade, and say, "Our students would like to program this in Java but if your system is in C++, we'll settle for doing it in C++ if you'll back off on some of these requirements that look like they would take more than two semesters to do." There's a bunch of horse trading a system like that will help you do.

Brock: So was that the case—it wasn't, it sounds like for this—and I want to loop back to DARPA, but—

Boehm: Yes.

Brock: —to follow this line of the WinWin systems, it sounds like it was less a system where you were modeling all of the different interest and constraints of the various people ahead of time, but rather a system where people could, it would surface all the various different interests and make people de-conflict in a way or something. Is that right?

Boehm: Yes. That was it. Also, we even found it was important for them to prioritize things. The students really didn't know how much they could do in architecting at the first 12 weeks and building it the next 12 weeks. Again, we had the COCOMO model that they could use to estimate how much effort it was going to take to do these. It basically says, "Here's your most likely amount of effort to do this, and here's the optimistic and pessimistic versions," which were basically 25 percent higher or lower.

Brock: Right.

Boehm: So we would say, "For what you've converged on that fits the COCOMO most likely model, take the things that fit within the pessimistic model and say this is this is going to be your core capability and then the team will guarantee that it gives you this." But if it turns out that things weren't, maybe were even

easier, they can fill up and with all the rest of the prioritized features. We found that prioritizing was another really important thing to do because when you went to inspections and testing pretty much all of the research and test technology assumed that every test case was equally important. What we found was that it wasn't. The high priority things are the things that you want to test first, and the things you want to inspect first.

Brock: Right.

Boehm: That way we got a lot of value. We found that, again, you've got twice as much value of defect avoidance per hour of inspections than…and that sort of thing. You see that happening with modern technology, that Amazon puts out a new version every 11 seconds?

Brock: <laughs>

Boehm: And you say, "How can they do regression tests over their millions of lines of code?" Well, they prioritize things. They only regression test the things that are really important and that way they can—there's a little risk to it, but the next 11 seconds they could work on the next one.

<laughter>

Brock: Well, I wanted to, if I could, going back to the time that you joined DARPA in 1989, there had already been this concentration or recognition that a process for creating software was important to DARPA.

Boehm: Mm-hmm. Mm-hmm.

Brock: I wondered was there sort of a project or a effort or some sort of event that everybody recognized as really underscoring the importance of software and software creation to the Defense Department, or was it an accumulation of an experience? I guess what I'm struggling to ask is, was there like one big thing that was iconic about software's importance or was it rather the accumulation of experience of the importance of software?

Boehm: Well—

Osterweil: You got to talk about the CMM [Capability Maturity Model] with Watts Humphrey. <laughs> I'm not sure if that's what you want to talk about, but the CMM with Watts Humphrey is boy, DARPA had a heavy role in that, don't you think?

Boehm: Yes. Well, basically, the three software intensive programs that I inherited or had been assigned to DARPA were the STARS program that was doing the environments, the Ada program and the Software Engineering Institute. One of the things that I was involved in at TRW in 1986, '87 was having TRW test out early versions of the Capability Maturity Model on our subcontractors. We gave them some pretty good feedback on some things and learned a lot from them on the things that they had.

The Capability Maturity Model was something that was a game changer for the Defense Department, particularly for the Air Force, which was the first organization to say, "If you're not Level 3 in the Capability Maturity Model you can't bid on this program." That got all the aerospace companies to immediately invest in just exactly what did they have to do to get to Level 3. Level 2 was basically the equivalent of blocking and tackling, that you had to understand configuration management and requirements management and test management and things like that. Level 3 said you had to institutionalize these in your organization and then you had to have policies and standards and education programs. And emphasizing a few techniques, like inspections and the like, that were really only a tiny part of verification and validation, but the SEI liked that and it sort of highlighted them.

It was very nice being the customer for the—and the manager for—the Defense Department of the SEI because its director was a guy named Larry Druffel, who I had known for a long time, both on campus and in technical things and non-technical things. We both were about equal as tennis players, and so each of us had a tennis court in our backyard and so when he was in LA, we'd play tennis, and when I was in Pittsburgh, we'd play tennis. And we played—

Osterweil: But he had the better backhand.

Boehm: I think so. <laughs> Yes. And yes, we played tennis in Singapore, and that was a fun thing and just all over the place, yes.

Brock: It sounds like there was just a growing recognition and a growing emphasis on the importance of software to all of these defense-related projects and activities and systems. So this, almost like this forcing function to have the entire community really adopt best practices for developing software.

Boehm: Yes.

Brock: Yes.

Boehm: And it was true within the DARPA Information Science and Technology office as well. Basically we had an AI program, which also included speech understanding, a natural language understanding and image understanding and things like that. We had networking, which was software intensive. We had high performance computing, which the hardware was really important, but the software was pretty important as well.

Brock: Right.

Boehm: We had a really spectacular system called SIMNET [simulator networking] that was a—one of the things I got to do as the office director was to go and visit things that were being implemented on these programs. One of them, the Army had adopted SIMNET. They had an Army base at Fort Knox, which besides having a whole bunch of gold there, it had a big building with 200 boxes and inside the boxes were something that looked exactly like an Army tank. And it had windows that were the same size

of the tank windows and it was engineered so that it could shake, rattle and roll like a like a tank and things like that.

What SIMNET would do is to train soldiers to become tank pilots and learn how to drive in formation or avoid obstacles or communicate with each other or things like that. Or determine yes, how could they defend themselves from attacks and that sort of thing. Again, this was a challenge to networking technology but again, a huge challenge to software technologies. Just about all of the programs, you know, we had some programs that were really generically addressing key software approaches. One of the program managers, Bill Scherlis and I said, "Yes, we've got to stop building software one instruction at a time. We really ought to do it one component at a time."

One attractive way to do this was to within particular domains. We had a program that was saying, you know, let's try to build components for command and control systems, for ground vehicles, for air vehicles and then things like that. As I remember, Dick Taylor got one of the these that took some of the Arcadia technology and used it in a communication and control environment, and tried to come up with components that would interoperate. You could compose different kinds of command and control systems for different purposes using these components and tailoring them. So, we had that.

Another thing that Bill Scherlis and another DARPA guy named Steve [Stephen] Squires got involved in was, this was about the first time that the Internet got hacked. People said, “We've got to come up with some way to build technology that avoids this and builds practices and a place where people could go to get help on hardening their systems.” Bill Scherlis and Steve Squires said, “We ought to get the SEI to do this.” And the SEI said, “We will create something called the Computer Emergency Response Team, or CERT.” It is still the biggest security organization in the U.S. government.

Osterweil: I was at Dartmouth visiting that day, that day that that worm got let loose on the Internet.

Boehm: Oh, really?

Osterweil: Yes. We had been—the Arcadia Project had been pulled into DC to stand and deliver, and to do a major presentation to Scherlis and Squires and I think Jack Schwartz was there, too. We spent weeks preparing and doing our slides and everything and we had it all planned out and we showed up and we got about two or three slides into it, and suddenly people started jumping up and running out of the room and buzzing back and forth and other people came running in and running out. We're like, "What in the world is going on?"

Boehm: <laughs>

Osterweil: "We've got this presentation here." And finally, at one point they said, "You guys are just going to have to go home."

Boehm: <laughs>

Osterweil: And then on the plane on the way home we heard about the infamous Internet worm—

Boehm: Yes.

Osterweil: —that ate DARPA. It was quite an amazing day to be there.

Boehm: I guess, right.

Osterweil: And obviously it had a lasting impact.

Boehm: Yes. I do remember.

Brock: That occurred while you were there also, Barry?

Boehm: No. That was when Jack Schwartz was running the office.

Brock: Okay.

Boehm: Yes. So that was '88 that the worm—

Brock: Right beforehand.

Boehm: Yes. Yes.

Osterweil: Yes. Just luck that I happened to be there.

Boehm: Yes. <laughs>

Osterweil: So David, did you wind up interviewing Watts Humphrey before he passed away about the CMM?

Brock: No, I don't believe that we have.

Osterweil: Ah. Well, that's an interesting story. If you and I are together this spring, maybe I could tell you something about the CMM, because I was on Watts' advisory board at IBM. Then when he went to the SEI he invited me to come and be on his advisory board there. Bob [Robert] Balzer and I were members of that advisory board right from the beginning all the way up to the mid-90s and could tell you a lot about the CMM and where it came from and how it evolved.

Brock: That would be—

Osterweil: And the reactions to the CMM, in the form of things like agile methods and those kinds of things were probably as significant as the CMM itself. Barry was there while this was going on at DARPA,

but I still remember conversations. Watts Humphrey was just this amazing dynamo of a man and some of what he wanted to do with the CMM seemed to most of us to be kind of wrong-headed. So you probably maybe never met Bob Balzer, and you and I haven't gotten to know each other very well and it sounds like you haven't met Larry Druffel, but between the four us, that would be Barry, Larry, Bob Balzer, and me, I think four forces of nature, and we could not budge Watts Humphrey one inch.

<laughter>

Osterweil: He did what it was he had in mind, and some of it was good and a lot of it was unfortunate, but it had an amazing reformative effect on software development. How people feel about process to this day is strongly influenced by what Watts Humphrey felt was the way it should be done. I don't know if you want to go into any further than that, Barry, but a lot of it happened on your watch.

Boehm: Well, it did start having strains, and it also had successes, in that fundamentally the system engineers were pretty chaotic. They said, "We need a system engineering capability maturity model." So, two of those got started with different system engineering oriented societies, and they were not all that compatible with the software ones. One of the things that I was just rotating out when they started to converge the Software Maturity Model and the Systems Engineering Maturity Model, which became the Capability Maturity Model Integrated, and displaced the two that were sort of confusing the situation.

Originally, they were really oriented toward this waterfall model that basically says, “Software is just as easy as having water trickle down a waterfall.” You start with requirements. You go to the preliminary design, detail design, code, test, integration, and acceptance. People found that this used to work for heavily scientific kind of programs, which was what TRW was doing in the '70s, was primarily things that were keeping track of satellites and keeping track of the information that the satellites were doing and the like. It was all batch processing kind of things.

All of a sudden, things became interactive processing and you really needed to do a lot of prototyping to do this. You couldn't say, “We are going to organize the screen like this,” and have that propagate through and deliver something that the commanders couldn't use. A lot of things said you've got to do a bunch of system understanding and risk reduction and an operational concept formulation and things like that before you start going into actually building stuff.

Brock: Mm.

Boehm: This was actually a good thing because the system engineers were more cognizant of that situation. A good example was that the first capability that you were supposed to worry about was requirements management. The first sentence in the CMM's description of requirements management, was “Requirements management and requirements definition is not the job of the software engineer, but a prerequisite for their work.”

Brock: Hmm.

Boehm: At TRW, that tended to be the case when people were creating a new satellite system. The propulsion people and the structures people and the guidance and control people and the communications people would sit around the center table with their architectures and define what the system was going to be like. The software people sat around the periphery of the room waiting for somebody to give them some requirements, but after a while the software was really driving the systems. In 1960, eight percent of the functionality of an airplane was being driven by software. By 2000, it was 80 percent. <laughs> So if you're going to lock the software people out of the room and put them outside the table, things aren't going to go so well.

Osterweil: The software's not going to work, and the plane isn't going to fly right and the software people are going to get blamed for it. So I worked at—

Boehm: Yes.

Osterweil: —Boeing in the late '70s and I can see this. It started out with the mechanical engineers laying out the frame and then you wrapped up with the propulsion people and after a while you had the electrical engineers stringing some wires, and then when they got done with all of their boxes and pieces of sheet metal they'd turn it over to the software engineers and they'd say, "You guys need to make it fly." And the software engineers would go, "Now?"

<laughter>

Osterweil: “As they roll up?” And then everybody would point fingers at the software people and said "Well, we did our jobs and these software people, they're helpless. They're hopeless. If the plane doesn't fly it's their fault." Everybody else had left the room by that time.

Boehm: <laughs>

Osterweil: It was clear that that was not going to last very long.

Brock: When did that sort of model change?

Boehm: Well, there were various things that changed. One was something that just as we had—one of the things that I did in the mid-70s was to lead an activity for TRW to have policies and standards. At the time, all the things that we had done well in the '60s and '70s is doing these high-tech kind of things, so the waterfall model was good. It was all batch processing, there was no user interaction or anything like that. We said, “We have the most effective way of building software that there is.”

There's a really outstanding engineer by the name Winston Royce, who in 1970 wrote a definitive paper for an electronics conference that described the waterfall model, but it did say that you really want to do some building it twice, so that you know roughly the directions you want to go. But basically our corporate publicity said, "You should come to us for software because we are the masters of the waterfall model."

Then when the prototyping stuff started coming along, Win Royce and I would have a bunch of conversations about, “What are we going to do about this?” Basically, looking at the statistical decision theory you can say that prototyping is a form of risk reduction, and you ought to have some risk reduction kind of things going on. Rather than doing a sequence of specifications, you want to be doing a combination of specifications and prototypes. A way to sort of look at this was to say that you were not going step-by-step, but you were going around a bunch of spiral circles. In a couple of—I can't remember, were you in the early [International Software] Process Workshops?

Osterweil: I missed the first one, but I was—

Boehm: Yes. So the—

Osterweil: —there for the rest of them.

Boehm: —first one was in England with Manny [Meir] Lehman and—

Osterweil: I missed that one.

Boehm: Yes. Right. Yes. But we hosted the second one in California, and I presented the spiral model and various people said "Hey, that looks like it's got some better characteristics than the waterfall." It started taking off for things that were more risk reduction kind of steps. Even that, though, I think was a little bit too heavyweight and well, it was good for big defense systems but it was heavyweight for business systems and the like.

In the mid-90s, you started seeing people doing agile methods, and saying that, “What you want to do is just interact with your customer to find out what they want and build them an initial prototype. What they don't like about the prototype, you do another sprint and fix that and say now what do you want.” And so, agile methods became a very popular thing to do.

So, there were some big battles between the agilists and the waterfall people who thought they had solved the problem by calling something a V-Model, and you go down this way, and you go to requirements, design and then you get to the bottom and you do code, and then you test against the design and you test against the requirements. It's just the waterfall <laughs> tipped over. But the maturity model was reluctant to change, and determine when to do the waterfall and then when to do agile. When the—

Osterweil: So, I think the truth is in my contact with Watts, he always claimed that the CMM was only about determining predictability. He was not particularly interested in how you build software, just whether your projections about cost and budget, scheduling and budget, could be trusted or not. His claim was, his motivation for the CMM, was just so that the Defense Department knew which of those lying contractors could be believed, and which ones could not be believed. If you got to Level 3 then you probably could make believable predictions. Then at that point, you're okay with Watts.

As to actually how you built the software he really—it was really hard to see it at first but he really—he backed away from that, sort of like, “Not my business.” But, in order for him to make his determination of whether you were Level 1, 2, 3, 4, or 5, you had to fill out these enormous amounts of paperwork, because he looked at these documents that you produced and decided whether they indicated that you were trustworthy with this side or the other characteristic. It was that paperwork that people resented. It was at that point that the agile methods community popped up and said, "All of this paperwork is for the birds. The only thing people really care about is they're running code. So what you should do is write the code first, and get to the paperwork later on."

Watts actually, as far as I can recall, never really felt that that was a bad thing except that it led to lack of predictability. He was pretty much agnostic about whether people used agile methods or other kinds of methods, as long as they were predictable. But that whole extreme programming and agile methods community sprang up as a response to Watts and his pushing for all of the paperwork it took to document what level you were supposed to be at.

Boehm: A classic example of that was a picture that Lockheed Martin made of one of their really big projects, “CMMI Level 4 Memorial Library,” which was a stack of shelves about as big as this conference room wall. They were all spiral binders of results of each code inspection and things like that so that—yes, the SEI also did training courses and licensed people to be auditors or lead auditors. If you wanted to be Level 3, you had to bring a team in that would audit this library and say “Yes, you're Level 3,” or “You're Level 4,” or whatever. The agile people made fun of all this.

I remember one other interesting parody of a call for papers for a conference that was called "Waterfall 2006," and it was going to be held in Niagara Falls and it was, “Agile people are not invited.” <laughs> There was really quite a battle between the agilists and the waterfall CMMI people. The SEI just really had a hard time figuring out what was the equivalent of Level 3 if you were doing agile, and the like.

Brock: Hmm.

Boehm: They were really behind the times even by 2006, sort of hanging in the waterfall.

Brock: Just following that line out, has that resolved to this day?

Boehm: I think what has been resolved is that there are combinations of agile things that you want to do, and plan-driven things that you want to do. Another book that I ended up co-authoring with a guy named Rich [Richard] Turner was something called *Balancing Agility and Discipline*. It basically said that there are things where lockstep discipline is not a very good thing to do, but there are places where coordinating what you're doing is a good thing to do. Showing some examples of where the “Agile Manifesto”—they had a manifesto, that basically said things like, “Embracing change rather than following a plan.”

This turns out to be a good thing in some cases but not a good thing in other cases. At one point at TRW, I was on a panel that was saying, “What were the causes of so many missiles getting launched from

Vandenberg and then blowing up because of the software?" In most cases, it was because people are responding to change over following a plan and saying, "We've got this fix that, we've got to do, or we've got this telemetry station that's moved and we've got to put a patch in the software. There's not enough time to do the regression testing and the configuration management and following the plan." And so, launched the rocket and boom, there it goes. Responding to change over following a plan may not be good in some situations.

Brock: <laughs>

Osterweil: So David, I guess what I would say is that things have started out roughly as follows. If you have a website or a web service or a browser or something like that that you want to get up pretty quickly, the community tends to understand that you would use an agile method, and you would get this thing up and it would work pretty well but probably would need to be adapted and adjusted and debugged and so on, and people understand about that.

So, you use agile methods to bring it up real quick and then use agile methods to make it better and better and better. What Watts Humphrey would say is, "Well I don't see a whole lot of predictability there." Because to borrow on Barry's example, sometimes the rocket blows up and sometimes it doesn't, and I want to be more predictable than that.

I think that that's quite correct, and in fact for things that really matter, if your whole bank is going to rest on this or your entire business is going to stand or fall depending on whether this thing works correctly or not, people don't tend to use agile methods. They may claim that they do, but they sort of in an agile way move down the waterfall, in some sense, or around the spiral.

Brock: Right.

Osterweil: So I think when things are important people really do fall back on plans and they want to know what they have, and they want to be sure they can trust it. That's a more ponderous but reliable way of doing things. But most of the work of the world is done with these agile methods where something gets brought up. You know version one is not going to be very reliable, or is not going to be very fast, but the next version will be better, and the world goes along with that. From my point of view, much of the problems that the world has is in thinking that more software will fit into the second mold of agile development without causing problems than is really the case. So now you have important things that are brought up with agile methods in a rather slapdash way and now you have big failures.

Brock: Hmm.

Osterweil: But I think it's resolved into a judgment which is made sometimes incorrectly, about whether to use agile methods, whether it's safe to use agile methods, or whether you really have to do the more ponderous and reliable thing.

Brock: I wonder how issues of computer security may change that back.

Boehm: Well yes, basically what the agile people say all you need is user stories. Let's say, as the manager I need a way of seeing an exception report if anything gets more than ten percent behind schedule, or I need a trend analysis that does that, and people will start building things that satisfy those stories. Unfortunately, the agilists treat security as a story, and say "We want our system to be secure." Well, we won't know what that means until we build some stuff. So, “Let's put security into release six.” By the time you released one, two, three, four, and five you’ve built some unsecurable software. People have come to recognize that you don't want to get into that trap.

Brock: Mm. I wanted to loop back to your story about rescuing the AI budget at DARPA.

Boehm: Uh-huh.

Brock: Did you come in with a predisposition toward the value of artificial intelligence work, or did that seem like such a big shift that you really wanted to—the de-funding seemed like such a big shift that you really wanted to take another look at that? Or had you had some prior experience with AI that shaped your thinking?

Boehm: Yes. Fundamentally, one of the advantages of the RAND Corporation was that during the summers they would bring professors in to complement the RAND staff in looking at new things. In the '60s, there were several summers where Allen Newell and Herb [Herbert] Simon would come to the RAND Corporation and really do pioneering work in AI. One of Simon's grad students was Ed Feigenbaum, and so he and I were about the same age and were sort of buddies and still are. So, I had a good appreciation, but another of my roommates at that time was Stuart Dreyfus, who was the brother of Hubert Dreyfus, who wrote the book *What Computers Can't Do*, and showed all of the failed predictions that say, “In 1958, in 10 years, the computer will be the world's chess champion.” Well, they got there but not in ten years. I was getting sort of a balance of skepticism and enthusiasm.

In one case, I was going from being just a programmer to being—I ran something called the Computer Systems Analysis Organization at RAND. One of the things that we wanted to make sure we did with the system analyses was to apply them to various challenges. One of them was the Minuteman command and control system. Basically, you had these missiles that were in bunches in Montana, North Dakota, and various places. People were worried about the fact that if the Russians targeted their missiles the right way they could disrupt the communication system so that the Minuteman wouldn't be able to launch.

Brock: Mm-hmm.

Boehm: About that time I think I mentioned the last time that my wife, Sharla, worked with Paul Baran on distributed computing, and I ended up doing a simulation of how that might work in a Minuteman command and control system, and finding that you needed a learning algorithm to determine if you had a system where parts of the system were going down, you needed some way to reroute things that weren't the shortest path until one of the thin nodes on the shortest path disappeared.

Brock: Right.

Boehm: I was doing this in concert with the people at UCLA, Len [Leonard] Kleinrock and the like, and came up with this learning algorithm that said if a message comes in, it says "I came from node 10 and I got here via node 14, and I got to your node 16, and I need to go to node 22, and I got here from there in three hops and it's going to take me two hops." So that's really information that's useful. If you get something that's going the other way you can say , “The best way that we know to get from there to node 4 was going out on node 10,” and that sort of thing.

Brock: Right.

Boehm: So I interacted with the AI people to see what are the best ways for this system to learn about that kind of thing. It was an appreciation that, yes, could be used for nice, practical systems and they had good algorithms for learning that kind of thing.

Brock: It seems like an interesting time when the move was being made at DARPA to defund the artificial intelligence research because this is, if my memory serves, this is the beginnings of the so-called “AI winter,” right?

Boehm: Yes, mm-hmm.

Brock: Where the great expectations for expert systems and that whole sort of enthusiasm around that was starting to abruptly <laughs> shift or something. Was that part of it? Were people…

Boehm: Yes. That was—

Brock: Was that part of it?

Boehm: —a big thing that, yes, some of the people that Lee and I were mentioning, like Bob Bressler, were victims of the AI winter. Ed Feigenbaum had founded this company called Teknowledge that basically said… Another thing that made it popular and then profitable early was the Japanese Fifth Generation initiative, where basically Ed Feigenbaum and another woman wrote a book that basically said, “The Japanese are going to blow us out of the water because they're going to harness AI and then we're going to be way behind.”

And Teknowledge was a bunch of smart AI people who could go to a company and then say, “Here's some things you might do.” So, Bob left ISI [Information Sciences Institute] and went to Teknowledge, and things went really well for a while. Then basically, the Japanese Fifth Generation thing sort of petered out and the AI things really didn’t cover the complexities of some of the things that the companies needed. So, yes, there was this AI winter.

Osterweil: But David, in fact there have been other AI winters. There's been at least two or three distinct AI winters.

Brock: Yes. <laughs> So I’ve learned.

Osterweil: And basically these are very, very smart people, and they're expert marketeers and they always over-promise. When I became a manager <laughs> long, long ago for the first time, my boss said "Your job as a manager is to manage expectations. Never let people's expectations get out of the box, because if they get out of the box you can never win. If you do those incredible things people will say, 'Well sure,' but most likely you're not going to be able to do those things and people are going to get mad."

Boehm: <laughs>

Osterweil: So the AI people have always made the mistake of over-promising. So yes, there were people who said, in 1955, "Pretty soon a computer's going to be the world's best chess player. Computers are going to automatically translate any language into any other language faster than anybody can even think the words." Said all this crazy stuff, and it's just inevitable that at some point these promises are not going to be met and then there's a backlash and there's an AI winter.

Right now AI's riding very, very high and you hear AI people making the same kinds of irresponsible promises. Some of them will be met, and a lot of them won't. My own personal belief is that just as every AI boom is bigger than the previous one, every AI bust will be bigger than the previous one, too. I'm about to write an op-ed piece that I'm going to send someone called "(Another AI) Winter Is Coming."

<laughter>

Osterweil: Because they're doing it again.

Brock: What's your perspective on that, Barry?

Boehm: Well, it's sort of similar with agile methods, that these things are really good for some things but they have their soft spots. For example, you can spoof an AI learning system. People have found that—and I think the first one was the AI system that did checkers and basically the way you beat it was to make really bad moves and anything it did caused them to win the match. So you'd play ten matches and then they'd say, "Well let's raise the stakes and play one more." The AI thing would capitalize—make bad moves that were being rewarded, and then lose the match. So, this kind of thing could happen in applying AI to security and that sort of thing.

Brock: But nevertheless, in that 1989 iteration of the kind of troubles with artificial intelligence research and activity, you decided to continue the course despite the downward pressure, if you will.

Boehm: Yes. Fortunately, while I was there were some big wins for AI. One of my program managers was a Air Force major when I got there, and he got promoted to lieutenant colonel, but he introduced himself as saying, "I am the major cross that you're going to have to bear."

Brock: <laughs>

Boehm: He's now the number two guy at Georgia Tech, and he was the director of the Software Engineering Institute and had a really outstanding career. He got some CMU [Carnegie Mellon University] and MIT people to come up with an AI constraint-based planning approach to solve transportation problems. This was in 1990 and '91, and they came up with a system that could do in four hours using constraint-based planning what it was taking the clunky transportation command software four days to do. Just about in 1991, we needed to get a half a million people off to the Middle East to fight the first Iraq war. And the transportation command said, “We're confiscating your Sun computers because we need your system to plan all of these things.”

Brock: Wow.

Boehm: They replaced them eventually. <laughs> But fundamentally, this was a key to getting all that stuff there really fast, and a triumph for AI. Steve Cross got the Golden Nugget Award from the commander of the Air Force and went on from there. So, yes, there were enough examples like that that you can make a case that AI was something that was really going to help.

Osterweil: It is the world chess champion, and the world Go champion.

Boehm: Yes. Right.

Brock: Yes. Absolutely. Well, maybe we could talk just a little bit about what is behind this report that you brought to the interview Barry, which sounded like a major effort of yours while you were at DARPA.

Boehm: Yes. One thing that we had really was a community of software research people. Some of them were sponsored by DARPA, some by the Army, Navy and Air Force Offices of Scientific Research. What I was finding was that things were underfunded, and that the professors didn't have enough funds to really make critical masses of grad students to scale up their approaches. What I proposed, and my DARPA director said, "We don't usually do things this way, but yes, if you want to convene all of these various Offices of Scientific Research and potential consumers of the technology and the like, and come up with a strategic plan for how to make software get a great leap forward."

So this rather thick thing went through all of the of the various technologies that were involved, including process technology, and requirements technology, and test technology and associated things that would apply AI and other kind of things to them. And coming up with a thing that said, "Here's the first 10 years of the plan, and here's what we would expect as milestones,” and things like that. “And here's a cost model, that basically says if you can speed up the process of getting from requirements to design to code to test and the like, here's what the savings are going to be, and here's what the cost of the program is going to be, and then you've got a very large return on investment."

People more-or-less bought the business case. We had this big event where we showcased this, and then got everybody excited about it. Then my boss got promoted from being the DARPA director to being the director of Defense Research and Engineering, and he said, "Yes, let's float this and see if it flies."

Unfortunately, what I found was that a lot of organizations didn't want money added to their budget that they didn't control.

Brock: Ah.

Boehm: I had this the two-star admiral who was the head of the Office of Naval Research. He said, "I get my money from the Chief of Naval Operations and I follow his priorities, and our big priority right now is corrosion. Our boats are getting corroded and we need more research in corrosion technology. And software, I can't really accept your software money. If I get more money I'm going to use it for corrosion." Again, you found all sorts of things in political Washington that complicated selling something that looked good for what you were trying to do, but not necessarily for what other people are trying to do.

Brock: Right.

Boehm: Who were higher ranked than you were.

<laughter>

Osterweil: Barry, you know, I think you're being modest. Basically, as usual, you were ahead of your time. I think it was a time when people had not quite focused on software as being the core problem—

Boehm: Mm-hmm.

Osterweil: —that everybody had that required a direct focus. And David, Barry is one of those guys who went to Washington and he had his head screwed on tight, and you knew what was needed, and he focused in pretty nicely on important software and computer technology and so on. What happened is exactly what Barry described as everybody said, "Oh, it's my problem too. It's my problem too." And everybody said, "Okay, then I'm just going to focus on it in my way." Instead of it becoming the focal point that Barry correctly saw it needed to be, it just got divided up into tiny little pieces. I mean at this point, there is this recognition and there is starting to be a kind of focus, but Barry's just ahead of his time, as usual.

<laughter>

Brock: So, your plan was never fully implemented.

Boehm: Yes, well, it didn't pass initially. Toward the end of my—yes, this was late '91 and I was going to disappear somewhere, and it was September, '92 when I left.

Brock: Right.

Boehm: As I was exiting, my boss was redoing the budget, which basically with the Berlin Wall falling and various things like that, a bunch of money that was being targeted for doing various kind of things with our

ICBMs [intercontinental ballistic missile] didn't need as much money as people were bidding for. And he said, “Here's $40 million, $40 million a year, and see what you can do to get to get this off and running.” Unfortunately, I only had three months to get it off and running, and the people down in the in the middle of the Office of Navy Research couldn't outvote their admiral in terms of—

Brock: Yes.

Boehm: —submitting a plan for doing this, and the admiral saying, "You're just complicating my life." <laughs> "I'm just going to use my Navy budget.” It looked like it was going to be a winner for a while, but it didn't make it. Yes.

Brock: Do you feel that the sort of needs that you had identified, the fundamental needs that this plan was designed to address, do they still exist today? Would a new—an updated version of this sort of plan be needed?

Boehm: I think one is needed. And I think it gets needed in getting companies—the Defense Department to build things more like what the really advanced software companies like Amazon are doing. Amazon comes out with a new release of their software every 11 seconds, and there's this approach called DevOps [development operations] that fundamentally says that you organize your system in a way that many things can be going on in parallel, many things can be integrated in parallel, many things can be prioritized in terms of rushing testing and the like. So that the most important things that need to be in there, the next 11 seconds are going to be there, and then the flexible plan for what's going to happen in the next 11 seconds, and then the next hour, and then things like that are being worked on.

Brock: Yes.

Boehm: There is a movement in Europe to and address this that… I don't know, Lee, have looked at any of our CERT talks, webinars?

Osterweil: I was <inaudible>, Barry.

Boehm: Uh-huh.

Osterweil: Barry, I think basically what happened, right around the time you were leaving DARPA was that the private sector came rushing by the government.

Boehm: Yes.

Osterweil: In the '60s, the '70s and the '80s, the government and particularly, the Defense Department, was the big dog in software. Basically, the Defense Department had all the big problems and spends all the big money and pioneered an awful lot of innovations and so on. And way back in the rearview mirror you could see the private sector and suddenly around 1992, ‘3, ‘4, ‘5, they came whooshing by at light speed. And suddenly they had way more money to spend than the federal government did. Somewhere,

by the year 2000, they really taken the lead and their agenda became the community's agenda. Their agenda was all about getting things on the Internet and getting things to market and getting things fast, fast, fast, fast, fast. Barry's agenda which said, “Hey, you know, it's important these things work—"

<laughter>

Osterweil: “Characteristics that we understand, and that we are assured of before the thing gets deployed.” All that stuff just, it just got lost in the dust of Google and Microsoft and Amazon rushing by putting stuff out there spending really orders of magnitude more than the federal government could. And, then eventually federal government turned around its nose pressed to the window and said, "Hey, could we have that stuff too?” And a lot of bad things comes out of that.

Brock: Hmm.

Osterweil: You have the Microsoft operating system that's built inside of lots of things that the Defense Department deploys that NASA puts out in space and it's just a cheap, easiest way, to sort of like, “Well let's just put that Microsoft operating system on and load a couple of applications on top of that.” And now you've got all of that dead weight of this operating system, which at this point at least it mostly works. But it's anyway—

Boehm: Yes, it used be the blue screen of death that you would find that.

Osterweil: Right, Barry tried to do the right but Washington's a very tough place. That I think, a lot of it got overtaken by events, and the events were, you know, the rise of this huge industry with, you know, unbelievable amounts of money to spend.

Boehm: Yes, I think another milestone and that was the Michael Cusumano and then Rick [Richard] Selby writing that *Microsoft Secrets* book that said basically, you want to carry multiple things along and pick and choose which ones are going to survive and that sort of thing. Unfortunately, they had this backward compatibility challenge, which really made things worse as things went along because they had to keep everything compatible with 3.1, 7.2, Vista, and all that kind of thing.

Brock: Right.

Boehm: Yes.

Brock: Going back to your comment about Amazon and updating the system every 11 seconds is that, literally, that updates every 11 seconds?

Boehm: Mm-hmm.

Brock: Is there a particular person or a particular group of people who are associated with this DevOps sort of technology?

Boehm: Yes. There's a guy named but Dean Leffingwell, who of built something called SAFe [Scaled Agile Framework], for “structured agile” or something. <laughs> It's proving to be pretty popular. There's this fellow at Chalmers University in Sweden, Jan Bosch, who has made a success of doing this kind of thing, and has got a consortium of big European manufacturing companies, like BMW and Volkswagen and Saab and Volvo and Ericsson and the like. They are finding that what Amazon can do for software intensive things they can probably do for cars. I don't see every 11 seconds they're going to beam something into your car's computer, but there is this idea that you're going to get continuous updates of your car rather than bring it in for service every 5,000 miles or whatever.

Osterweil: But Barry, I think the point is there's a person that’s involved—that's making this decision every 11 seconds in software.

Boehm: Yes.

Osterweil: And I guess it's not unfair, Barry, to say it's AI. So there's some AI kind of software that's making this decision—

Boehm: Mm-hmm.

Osterweil: —and deciding when it's safe and when it isn't, and have it put it in and when not to and when to pull it out.

Boehm: Mm-hmm.

Brock: Interesting. Well, maybe in the time we have remaining I would be interested in your reflections on two different things. One is about your decision to teach after your stint at DARPA, and what you have found to be the most rewarding dimensions of that, of your career at USC. I'd be interested to hear you talk about that. And also, then just to your perspective on what has been consistent and what has changed <laughs> in just the whole world of software, and where you see that balance of continuity and change heading in the future.

Boehm: Oh, okay. Sure.

Brock: Yes.

Boehm: Yes. Okay. Well, in terms of, let's see, what was—?

Brock: One was about coming to USC.

Boehm: Okay. Yes. At one point I was asked by Simon Ramo, who was the “R” in “TRW,” to help the company recruit more really bright software people because that was really a limiting feature on many of the things that the company was trying to do.

Brock: Mm.

Boehm: He asked me to go and try to come up with a program where TRW would hire some really bright people, and give them two days a week to go to either USC or UCLA to get a master's degree. This would be a plum thing that really smart people would do. Basically, I went around to the people at the UCLA and USC and surprisingly, they wanted to do it. For one, they were having a hard time hiring U.S. citizens. For another, Simon Ramo said, "If you're willing to do this I'm willing to give your department chair a nice big annual gift."

We came up with an initial set of things that would be part of the program and Simon Ramo said, “We don't want to get super programmers out of this. What I found in our company the most valuable people we have are T-shape people. They're very deep in whatever their specialty is. Maybe it's astrodynamics and maybe it's electronics, maybe it's communications, but maybe it's software. But the ones that are really helpful are the ones that know a little bit more about the other parts of it, about the management and the economics and all the other things.” This was the basically true of most of the new hires that we would get as software engineers at TRW. It would take a year to get them to appreciate and understand what configuration management was all about, what requirements engineering was all about, how do you estimate costs of things and the like.

And so, when I was 55 and saying, "I'm going to early retire or stay here for another ten years and get excused at 65, I think what I'd like to do is go to either UCLA or USC and take these I-shape bachelor's students in computer science and make T-shaped people out of them.” The software projects were a good example of that because they had to learn the requirements are things you negotiate. They aren't things that you just write up and follow.

Brock: Right.

Boehm: And that estimating the cost is going to be critical to whether you can do it in 12 weeks in the fall and 12 weeks in the spring. Basically we put together, not just the project course, but a master's in software engineering that meant that you—the core courses were not just architecture and things like that, but they were courses in human computer interaction and management and economics and things like that.

This turned out to be—worked out pretty well, because particularly when the students showed up for a job interviews, that basically they knew what configuration management was all about. They had a portfolio. They could show you the project that they did, and how they used all these various techniques and the like. We got a lot of more companies to come to the job fair and in particular look for the students who were going through this program seeing if they could hire some of them.

Brock: Right.

Boehm: So, yes. That was put off for three years by this surprise offer to be the DARPA office director. So in '89, I left TRW and '92 I came to USC, which—

Brock: Right.

Boehm: That was sort of the main the thing that I decided I wanted to do here. Doing the project courses meant that you had a critical mass of data that you could use to test out hypotheses that said prioritizing the inspections and the test will give you a payoff and the like. So we've had 15 PhD students who have used the project courses as places where they can—

Brock: <inaudible>

Boehm: Yes. Right. And say that this is not just a good idea. We've got data that show that it works, that sort of thing. It's been a very nice combination of education and research that comes out of this.

Brock: Right.

Boehm: That was item one, item—

Brock: So the other item was just, from your career's length association with the world of software, I was interested to hear your reflections on what fundamentals do you see as being consistent across that time with respect to the nature of software and the nature of making software? And what have been fundamental changes, and where do you see those going? What seems sort of it intrinsic, and what seems to change?

Boehm: Yes, I think the main thing is that change is going to be constant.

Brock: <laughs>

Boehm: We tell our students that, “Yes, where we're going to try to educate you to have a successful career out into the 2050s. We have no idea what it's going to be like there.” <laughs> We give assignments that say, “We want you to not just learn, we want you to learn how to learn.”

So the assignments in the Software Management and Economics course, say, “You are the CTO [chief technical officer] of a 500 person software company and your chief executive officer is a concerned about AI, or DevOps, or Artificial Intelligence of various kinds and the like. What you need to do is to give him an analysis of how mature are these, and what are their strengths and what are their weaknesses, and what would we have to do to address these. And that you're going to get graded on how incisive your analysis is, plus the number of different ways that you learn about things. So you can't just go to Google and stop there. You should try to interview some people that are in companies, or are developing this kind of research and things like that. You should look at the proceedings of conferences and the ACM [Association for Computing Machinery]/IEEE [Institute of Electrical and Electronics Engineers] literature, and so the more sources that you do the better your grade is going to be.”

We're trying to get them to learn how to learn, and that kind of thing. We've found that we have to learn how to learn in running the course.

Brock: Right.

Boehm: In 1995, when we did our first set of project courses, basically the students had to program their own Web crawlers and search engines and things like that. In five years, basically there were COTS [commercial off-the-shelf] products that did these. Every year we would find we'd spend our summer reinventing the course.

Brock: Right.

Boehm: Between 2000 and 2010, things went from being COTS intensive to sort of cloud services intensive, and Web intensive, and things like that. Again, it's continuous learning for the instructors as well as the students.

Brock: Right. I guess the last question that we have time for is, earlier in our discussion today we talked about the functionality of an aircraft, from going from eight percent that software was responsible for to whatever it is, 80 percent or higher to that.

Boehm: F-35 is saying 90 percent.

Brock: <laughs> Yes. Okay, eight percent to 90 percent. It also seems that from just all of the kind of systems and services of our society, the role of software has gone from eight percent to 90 percent, or—

Boehm: Yes. Yes.

Brock: You know, do you see that trend just going to continue? What do you see as you look out ahead?

Boehm: Well, that was… Lee, I sent you the ICSE 21 thing? The biggest conference in software engineering is the International Conference on Software Engineering, and it asked Lee and me and various other people to comment on the previous ones of these and what were the highlights and the like. I ran the one in 1999, and the theme was, “Preparing for the Software Century.” Here, 18 years later, who are the top five companies in the world in terms of market value? And they’re Amazon, Microsoft, Google Alphabet, the Facebook and one Chinese company, Tencent. Again, the software century is already upon us. <laughs>

Brock: Right.

Boehm: Where it goes from here is going to be a continuous learning kind of thing. I really don't believe that there's going to be a convergence in 2047 when the computers take over the people and the like, but there are people who actually believe that. Again, I think learning how to learn and keeping your mind open to new things coming along, that—it's amazing how many things you don't need any more.

<laughter>

Brock: Well, thank you very much. I think we'll need to break here. But I've just enjoyed it very much, so thank you once again.

Boehm: Oh, I enjoyed it very much. Thanks, Lee, for all your—

Brock: Thank you so much, Lee.

Boehm: Yes. Yes.

Osterweil: Thanks for inviting me, and I really enjoyed it.

END OF THE INTERVIEW

