Oral History of Barry Boehm, part 1 of 2

Interviewed by:
David C. Brock
Hansen Hsu
Lee Osterweil

Recorded November 14, 2017
Mountain View, CA

CHM Reference number: X8388.2018

David C. Brock: This is Tuesday, November 14th, an oral history interview with Barry Boehm at the Computer History Museum. I'm David Brock. And I'm joined by Hansen Hsu here with me at the museum, and Lee Osterweil is joining by a video link. So Barry, in doing a little bit of background research I saw that you were born in 1935, but I didn't see where. Could you tell us a little bit about where you were born and where you grew up?

Barry Boehm: I was born in Santa Monica, California, at a hospital called Mary Martin's which is now a Toyota dealership.

Brock: <laughs> And can you tell us a little bit about your family of origin, your mother, your father, the family you grew up in.

Boehm: Okay. My parents grew up in and met in Cleveland, Ohio. In the early '30s, Cleveland, Ohio, was not a great place for jobs and things like that. They decided that they would trundle out in a car and drive out to California where it seemed like there were better job opportunities. Santa Monica was sort of the corporate city for Douglas Aircraft. My dad got a job basically assembling airplanes for a while. Eventually he got into middle management and purchasing and things like that. But yes, it was pretty much of a company town. All of the bus routes could get the workers from wherever they were living in Santa Monica to the plant. There's a theater in Santa Monica called the Aero Theatre and it basically was put there by Douglas as a sort of an entertainment for the workers.

Brock: And had your father had a technical background before that? What had been his interests?

Boehm: Not really, no. He was good at baseball, bowling and a whole bunch of things like that. Very, very friendly, easy to get along with person. Did well as far as I could tell with his career at Douglas Aircraft.

Brock: It sounds like the company was both a big part of the community there in Santa Monica and also your family. Was that a big part of your youth growing up?

Boehm: Yes, quite a few of the classmates had their parents working for Douglas. When I got to high school one summer I got an intern job at Douglas bucking rivets which basically you sit inside the wing of an airplane and somebody pushes a rivet at you through a hole and you put a piece of metal up against it and they go rat-tat-tat-tat-tat. <laughs> It was a pretty noisy thing to do.

Brock: <laughs> You were the bracing for what they were riveting against.

Boehm: Yes, right. Yes. Basically, after you hit one tap then it was a good solid rivet. If it was two, you had to drill it out and try again. That sort of thing.

<laughter>

Boehm: The next summer I was a page at the Santa Monica Public Library, which was a much more quiet kind of thing to spend your summer doing.

Brock: And did you attend the public schools in Santa Monica?

Boehm: Yes. Uh-huh.

Brock: What was your experience of school like?

Boehm: It was basically a good time. My main sport was tennis, and we had a good tennis team that won championships. The tennis team was the main friends' kind of thing. School was—there was one math teacher who was particularly stimulating. His name was Bob Crawford, and he was a very nontraditional teacher. He would illustrate things with gambling kind of stories and problems. He went well beyond just doing algebra and geometry and trig and getting you into precalculus and that kind of thing.

Brock: Were there opportunities with Bob Crawford to kind of do more math outside the classroom? Was there a math club or anything like that that you took advantage of?

Boehm: Not really. But he would give me problems to try to solve, and so that was pretty good.

Brock: Did you have an interest in some of the other activities or pursuits that many times are associated with interest in mathematics, like puzzle solving or music? Were either of those part of your interests?

Boehm: Yes, I went through the traditional 'your mother wanted you to practice the piano.' I made a deal that I'll do it for two more years if I can then go off and play baseball.

<laughter>

Boehm: But yes, I still play the guitar. It's fun to have that as sort of a way to relax.

Brock: And we mentioned your summer work. I guess that was during high school. Since you were growing up, and I suppose into the earliest part of high school, perhaps, the Second World War must have been just—well, no, I guess it's before high school—just an experience for you growing up, especially in Santa Monica. What impressions did the war have on you?

Boehm: Well, it made us concerned. A bit up the coast and in Santa Barbara there were actually some Japanese that shot at the various buildings. The Douglas factory was turned into something that looked like a residential area from above.

Brock: Oh, I didn't realize that.

Boehm: They had some things called barrage balloons that if the airplanes came they would get caught in the cables of the barrage balloons. So yes, you know there was very well a war going on. Yes.

Brock: In addition to your involvement with the tennis team and with your summer work, did you have any other primary hobbies or activities? Were you a big reader? Was there anything else that stands out?

Boehm: Yes, I read a lot of things. Particularly, I read a lot of science fiction.

Brock: I was just going to ask that.

Boehm: Yes.

Brock: Can you tell us about your interest there, what sorts of things you were reading?

Boehm: Yes, well my gods were people like Isaac Asimov and Robert Heinlein and then a whole bunch of other authors. Ray Bradbury particularly lived in the Santa Monica area. One of the neat things about being a library page is that people like Ray Bradbury would come in and say, "Could you go down and get me this copy of the 1948 *Santa Monica Evening Outlook*." Or Christopher Isherwood would come in. There were a bunch of interesting people in the community that you bumped into.

Brock: Oh, wow. Were science fiction authors that you followed doing local readings or anything like that that you participated in? Or was it mainly through library work?

Boehm: It was mainly through the library work. Yes. But I really believed that we were going to go to Mars and then colonize it and all that sort of thing.

Brock: Is that a reading interest that has stayed with you?

Boehm: It did for a while, and then the more I got into things like the space world, and people who started reporting things from Mars and Venus that said they're hard to get to and they don't look very friendly. <laughs> I really still read those things occasionally, but it's more of a minor thing now.

Brock: And were computers at all in your consciousness through either direct experience or science fiction by the time you were ending high school?

Boehm: The one stimulating thing that did orient me in that direction was that the RAND Corporation is also in Santa Monica. They had a couple of high school days where you could go and talk to RAND mathematicians or see some of the RAND computing equipment. They built their own computer called the JOHNNIAC [John v. Neumann Numerical Integrator and Automatic Computer] which is sitting up here in the Computer History Museum right now.

Brock: And you took advantage of that and saw the JOHNNIAC?

Boehm: Uh-huh. Yes.

Brock: Would you say that it made a big impression on you? I mean it certainly—you remember it, but was it particularly…

Boehm: It was just sort of a memory but, yes, we'll get to things a few years later. Basically, my parents divorced and my mother and brothers and sisters all went down to San Diego. So, I lived there for a while, and worked for what was called Convair and then became General Dynamics. At that time there was no University of California at San Diego.

After graduating from Harvard—well, I interned as a programmer at Convair after my sophomore and junior years. It was sort of the logical place to go to work afterwards. It was stimulating, but basically I wanted to go somewhere where I could do research and get a PhD, and UCLA was sort of the logical place to go. So, I went up to Los Angeles and I interviewed at RAND, System Development Corporation, and Ramo Wooldridge. Since I had that impression of RAND before, I said, "This is really a neat place and it's in Santa Monica, and they are willing to let me have flexible hours so I could commute up to UCLA and take courses," and that sort of thing.

Brock: Just to step back just a little bit I was curious just about how you—the story of how you came to go to Harvard. Was that one amongst a number of places that you were looking at? How did that come about?

Boehm: It was really more diversity than specialization and technical things. I got accepted at MIT and Caltech and Stanford and Harvard. At one point I was very interested in going to Caltech because, again, they had these high school summer things and there were a couple of lectures by Richard Feynman that were just super stimulating. I thought, "This would be a great place to go."

But I was really curious about what life was like at the other end of the country. Harvard had sort of the number one person in anthropology and history, religions and a whole bunch of things besides math, plus having some really super mathematicians. And they had one of the best astronomy departments. One of the nice things was the first year and a half I was there, I had to get a student job to make ends meet. I was basically part of the kitchen crew. But at one point I took an astronomy course in my sophomore year and followed up with some of the professors and said, "Is there any possibility you would have any jobs that I could do rather than wash dishes and that kind of thing?" They did, yes, need some people that could help out in some of the technical support things that they needed.

Brock: Do you think it was your interest in an aptitude for mathematics, in particular, that explained getting into that roster of schools? That's quite a good list.

Boehm: Yes. And, again, with the science fiction, the astronomical observatory was the place you could get familiar with the planets and all of that kind of stuff.

Brock: So then it was returning—you combined returning to your mother's household in San Diego in the summers with this position at Convair, General Dynamics.

Boehm: Yes. Mm-hmm. Yes.

Brock: That was, I think, if my memory is serving correctly it was 1955, that was your first kind of programming job. Could you paint that picture for us, what that was like? How did you get the job, by the way, the summer job?

Boehm: Fortunately, Harvard has a alumni organization. If people wanted a job, in a particular area, there were alumni who could help you get in. The first, after my freshman year, I was a draftsman at the Naval Electronics Lab and basically drafting pieces of ships and things like that. That was interesting but not a career.

The next year they said, “We can get you a job at Convair.” They do airplanes and rockets and things like that. I went into the personnel office and said, “Well, let’s see what kind of jobs we have. What is your major?” Well, my major is mathematics. So he said, “I’m looking at my blackboard over there and it has propulsion and thermodynamics and digital computer lab. Well, digits are numbers, aren’t they? So why don’t you try that out?” Up to then I was wondering what a math major did as a career. Being an actuary or a statistician or things like that didn’t seem like really interesting stuff and wouldn't get you to Mars or anything like that.

<laughter>

Brock: So what was that laboratory like at Convair?

Boehm: Again, they had one of the most powerful computers that was available at the time. It was a UNIVAC ERA 1103 [UNIVersal Automatic Computer Engineering Research Associates 1103]. It was about the size of this room. So I went over, and I got introduced to the supervisor who walked me into this computer and said, “Yes, this is what we do here. We write computer programs that help us analyze rocket trajectories and rocket stability analyses and structural analyses and things like that. If you’re going to be here, there’s one thing I want you to keep in the front of you mind, which is that we’re paying this computer $600 an hour, and we’re going to pay you $2 an hour. And we’d like you to act accordingly.”

Basically, hardware was king. This introduced you to some really good habits that you’d want to manually execute your program before you put it on the $600 an hour computer. You’d like to make sure that you had very thorough test cases. You tried to build programs that were extremely efficient and didn’t chew up too many of these $600 hours in computing things.

Brock: You mentioned, too, kind of manually execute the program before running it on the computer. That sounds like a very painstaking pencil and paper exercise. Is that correct? What was that like?

Boehm: Basically, yes, you would put in some inputs and you would manually execute the program. In some cases, it was a pretty messy thing. My first program was to provide them a subroutine that could compute arc-tangents. And about the only way you could do this was by using a series that says, “The arc-tangent is A times X plus B times X squared,” and the like. Manually executing those until it converged was not a lot of fun.

<laughter>

Lee Osterweil: Barry, what language did you write that program? Was that FORTRAN? Or was that assembly language?

Boehm: That was really assembly language. Yes. The input media was paper tape at the time. The other that thing you could do in terms of debugging your program was that the memory was an electronic tube. Fundamentally, it had 4K of memory. When your programming was executing you could watch the places where it was executing, and if it started burning up a hole in the display you knew it was in a tight loop and you better turn it off and try again.

<laughter>

Brock: In talking to some other people working in software in this era, programming in this time and after, several people have referred to debugging also not just by visual inspection of the memory tube but by sound. Just the sound of the machine running, you could distinguish between the typical operation of the program or if it was getting into trouble of some kind. Was that part of your experience at all?

Boehm: Not for the ERA 1103. After this—

Osterweil: It was just a little transistor radio on the console. And somehow the transistor radio picked up static or something like that, and actually you could hear it but you had to put a little transistor radio there in order to receive things.

<laughter>

Boehm: There were clues like that. Yes, when I went back to Harvard I started taking—they were really called numerical analysis courses. But you had to program things to do them and execute them on—you know, they had a UNIVAC I and then they had the Harvard Mark IV. The Mark IV was something that Howard Aiken had done the Mark I, II and III and it was better than those. But fundamentally, programming was saying, "Take this particular storage area and then open up this wire and then push those bits down this wire until it gets to this other location here." In those cases, yes, you couldn't see what was going on but you could tell a little bit in terms of hearing what was going, on in terms of—

Brock: You could hear the relays in one part of the machine going—

Boehm: Click, clickety. Yes.

Brock: —and then in another part.

Boehm: Yes, that kind of thing.

Brock: Interesting. So we're talking now around 1956. Was the activity over on the MIT campus with Project Whirlwind or any of that? Did you have exposure to that, or know about that?

Boehm: Not really, no. I knew MIT was over there but there wasn't really that much—there wasn't a Cambridge computer club or something like that.

Brock: Right. And the work on numerical analysis with the Mark IV—that didn't put you off of your interest in programming and computers. It seems like you had turned to kind of focus in this direction.

Boehm: Yes. It was something that, yes, you could see that you were doing useful things with it. Actually, I almost put myself out of a job in that I was doing this work at the Harvard Observatory. Harvard had two telescopes on two mountains in New Mexico. They had a camera with some rotating shutters. So, if a meteor came down it would have a particular dotted line from here and a particular dotted line from there. What you could do is take the photos and reduce the traces to numbers and go through a bunch of calculations on a Friden calculator that could tell you how high was it when it entered the atmosphere, and how fast was it going, and how long did it last. I did this really for the professors. But they let me coauthor a paper. So my first technical paper was called "Photographic Lyrid meteors."[1]

Brock: I'm not sure I followed how you almost put yourself out of a job. Did you program that problem?

Boehm: Yes. I decided pushing the Friden was—if the computers could be available—and the observatory did have access to the UNIVAC I. Basically I created a program that would do all of the calculations. I still had to measure the things on the photographs and put those data into the computer program. But I didn't have to punch the computer all. They found other useful things for me to do which was good.

Brock: When you return to General Dynamics in 1956 was this also working essentially to program mathematical routines for the kind of engineering simulations that were going on?

Boehm: It was more and more, it was—this was in this era of, "Are we ahead or behind the Russians in intercontinental ballistic missiles and satellites?" The U.S. government basically had three 'money is no object' intercontinental ballistic missile projects. Convair/General Dynamics had the Atlas, Martin had the Titan, and Boeing had the Minuteman. The first two were liquid rockets and then the Minuteman was a solid rocket. Basically, yes, there was lots of money to do things, lots of computing equipment to invest in, lots of computer programs that people wanted to do a better and better job of getting more and more

[1] Wright, Frances W., Luigi G. Jacchia, and Barry W. Boehm. "Photographic Lyrid meteors." *The Astronomical Journal* 65 (1960): 40. Added by Frances Corry on August 26, 2018.

accurate engineering analysis of rocket vibrations. Or there were certain situations that if the rocket didn't go fast enough the pumps would cavitate, so you had to simulate what was going on with the pumps and the liquid, that kind of thing as well.

Brock: Were there also, in the Convair computing laboratory you worked in, were there also electronic analog computers were you using? Or were you only using digital computers?

Boehm: No, I got to wire a bunch of plug boards. It was hybrid computing at the time. But there was a rivalry between the analog people and the digital people saying, “You guys can’t integrate digitally. We can do that automatically,” and that sort of thing.

Brock: Was that really kind of two camps in computing? Was there a real analog/digital divide would you say?

Boehm: Well, because some of the problems required you to use both there was really more of a friendly kind of thing, but there were all of these ‘push your button’ kind of things that would say, “You can’t do what we can do,” and that sort of thing.

Brock: Right.

Osterweil: Barry, somehow I recall that analog people thought that they can handle larger variations in quantities in numbers. And I never understood that. So they conceded that digital computing was more accurate. It was precise. But that it couldn’t handle variations and the sizes of numbers. I just never quite got that. Do you recall that?

Boehm: Yes, I don’t recall them saying that. The digital was certainly more accurate than the analog was. But yes, for certain kinds of situations where there were complex interactions among the various things that were being simulated that the analog could do a better job than the digital because the digital would have to make assumptions about, “Is this a steady state, or an anomaly?” or things like that.

Brock: When you were finishing up your undergraduate studies had you been talking with Convair about a permanent position with them after you graduated? How was that unfolding?

Boehm: That was unfolding—well, yes. Basically, we had one problem with my family. A couple of people got sick and that winter I went home to try to help out and then did a little bit of programming at General Dynamics. At one point, I was about to say, “I’m not going to be able to finish my senior year.” The people in the computer lab said, “Hey, don’t do that. We’ll pay your salary or your tuition or whatever.” I said, "You don't have to do that. But I agree I should really go and finish this." But fundamentally, yes, they basically said, "Yes, we want to have you here permanently." So it was good.

Brock: Wow. A very generous offer.

Boehm: Yes. It was a really nice thing for them to do. It was a very interesting bunch of people. I had a carpool, and one of the people that was a programmer was Bob [Robert] Price who became the CEO of Control Data. Another one was Don Parker who lives up here and was one of the early people in computer security and computer crime and that sort of thing. It was right at the beginning with all of these people growing up and then filling interesting positions.

Brock: Was it the case with other members of your carpool—were these people who happened to be from Southern California? Or had they come to Southern California to work with computers?

Boehm: I'd say most of them were Southern Californians. Eventually, yes, the demand for programmers got so high that—this was more later when I left RAND and went to TRW [Thompson Ramo Wooldridge Inc.]. RAND was sort of a steady state, 1000 people on the staff. And TRW was getting bigger and bigger. They would go back to Wisconsin and Minnesota and show them all of these pictures of what life was like on the beach.

<laughter>

Boehm: So we did get a lot of people from the Midwest.

Brock: Let's see. When you took the job with Convair/General Dynamics, what was your initial role? Was that more of the same sort of work on programming mathematical routines? Or how was it changing?

Boehm: Well, I got into a group that was doing the biggest computer program that we had, which was a simulation of the Atlas missile doing various different things. In this particular case, after various kinds of experiences of people coming and asking about doing things with different aspects of the inputs. So, if you had different versions of the rocket nine engine that was being put on the Atlas missile, there were curves of thrust and fuel flow versus time that were varying at times. Sometimes, they wanted the rocket nine A2 and sometimes they wanted the A3.

What the engineers had done was to organize something that created several options for the key things that the rocket simulation needed. Fundamentally one of them was propulsion. So you could say yes, we're going to do constant thrust and fuel flow. Or you're going to do a rocket nine, A2 or A3. As new ways to compute thrust and fuel flow came along they just became another option that you could use.

Similarly, for aerodynamics, that basically you needed to have models of atmospheric density and pressure so that you could take—compute lift and drag and all of that sort of thing. It was really impressive to see how they had really made life a lot easier for themselves by organizing this into sort of a modular kind of approach which was… When I got to RAND and did similar things I was able to capitalize on what they had done.

Some of the assignments were really stimulating things. A little later after Sputnik, the Atlas was chosen to be the rocket that put astronauts into space. One of the things that they wanted to see if it would work was to put an escape tower on top of the rocket so that the astronaut would be in the escape tower. If something started going wrong with the main rocket that they could hit a button on the escape tower and then it would get away before the thing blew up. So I had the job of modeling the escape tower and its thrust and fuel flow and aerodynamics and guidance and control and all of those sorts of things and making sure that if they hit the button it wasn't going to turn over and get hit by the rocket then. But fundamentally yes, the computations indicated that this was a feasible thing to do. They said, yes, let's put those on the Atlas missiles and fortunately nobody ever had to use one.

<laughter>

Brock: While you were doing this work, were you interacting at all with people doing the same sort of work in these other missile programs? I can't remember when it started, but I think at some point, the MIT Instrumentation Laboratory gets going on Polaris. Maybe that's a little bit later.

Boehm: Mm-hmm.

Brock: But then, the people working on Titan and Minuteman. Was there kind of a group of people doing this kind of computer work who exchanged ideas?

Boehm: It's sort of interesting that, yes, things like the ACM [Association for Computing Machinery] were starting to happen. But the first professional society that I joined was the American Institute of Aeronautics and Astronautics. Basically, I was still thinking we'll get to Mars somehow. The NAA [National Aeronautic Association] had a bunch of monthly meetings and was hosting a bunch of conferences. As a junior member, I would get the job of meeting senior rocket scientists at the airport and driving them to the conference or things like that. It was a really stimulating thing.

Just about every big aerospace company had its German rocket scientists. General Dynamics had Krafft Ehricke, who was, again, big on doing the computations of going to Mars and back. I got to interact with him and this was, again, a really stimulating kind of thing and kept you thinking that, yes, maybe we'll get there after all. <laughs>

Brock: Was that also a meeting ground where computer people in these aerospace projects could interact with one another specifically, as well with the broader community of aerospace types?

Boehm: Well, that's one thing that I stumbled into was that, yes, like ACM and IEEE [Institute of Electrical and Electronics Engineers], they had special interest groups, and there was not a special interest group for aerospace computing. I was in touch with some of the leaders in the NAA and said, "Do you think we could do something like this?" There were some detractors that said, "Well, we don't have a special interest group for slide rules. I don't know why we'd need one for computers."

<laughter>

Boehm: But fundamentally, what they asked me to do was to connect with some of these people that I would meet at these conferences and put together a special issue of the society magazine on aerospace computing. We got some really good people to do that. One of them was Herb Grosch. I don't know if you know Grosch's law, that basically—

Brock: I don't. I'm sorry.

Boehm: —says that the—how does it go. It was basically that it was an early version of Moore's law, that basically said that you only had to pay a square root for your next generation computer or something like that.

Hansen Hsu: Wasn't Grosch's law sort of—similarly to Moore's, I guess, it was economic in nature, but it seemed to imply that computation was more efficient concentrated in large mainframes rather than distributed into smaller computers.

Boehm: Yes. He was really a big mainframe guy. He was with General Electric at the time, but we got him to write an article in the special issue and a whole bunch of other people put together some very interesting things. And so, the society said, "Yes, this looks like a viable thing to have a special interest group in." So we went off and did that.

Brock: Just reflecting on your comment about each of these big aerospace companies having their own German rocket scientists, and the person you interacted with having the Mars visions and the rocket to Mars.

Boehm: Mm-hmm, mm-hmm.

Brock: Unless I'm mistaken, I believe that was a big theme for [Wernher] von Braun.

Boehm: Mm-hmm.

Brock: I wonder if that German rocketry community, if that going to Mars with a rocket was a real common motivator for them.

Boehm: Yes. Wernher von Braun fundamentally didn't want to—as Tom Lehrer put it in his Wernher von Braun song, said, "I make them go up. Who cares where they come down? That's not my department, said Wernher von Braun." But he was really more interested in conquering space and the like.

Brock: Interesting. What was the reaction to Sputnik going up in the aerospace community in Southern California? What was that reaction like?

Boehm: It was weird, basically. Yes. As we were going back and forth in the carpool in the winter when it got dark early, you could look up and there was this thing going around and—"That's not ours."

<laughter>

Boehm: "It's a Russian Sputnik." Again, yes, the money faucets were turned up and people said, "We've got to get there too."

Osterweil: So Barry, my recollection was that they wanted physicists and chemists and mathematicians, and you didn't hear a whole lot of clamor for people who knew things about computers. It was before somehow there was a really good recognition that computers were central to all of this, as I recall.

Boehm: Well, yes, the kings at the big aerospace companies were really the rocket scientists. The computer people were helping them analyze the problems that they encountered in terms of…. The Atlas was structurally impossible in that it had to be full of fuel to be able to escape being collapsed. So, you want it to be as light as possible. Having something that was sustained by the fuel inside it was a good thing. But, yes, there was one classic example of somebody doing a test where they were taking a rocket, an Atlas rocket on a truck across the desert and somebody took a shotgun and shot a hole in it and it went, “woomph.”

<laughter>

Boehm: It really did need all that stuff inside it to make it viable.

Brock: In addition to the use of the digital computers and software for simulation and engineering research and design, did you see or did you have exposure to any beginning use of the computer for the other aspects of these missiles programs, like keeping track of the budget or keeping track of the organization of the project, or inventory, things like that, the more social organization of the projects?

Boehm: It was such a big company that, yes, I didn’t have much connection with that. It was really only when I got to RAND Corporation in 1959 that basically it was a thousand people, but 200 of them were engineers and then 200 of them were economists and a whole bunch of physicists and cost analysts and things like that. In that case, you really get exposed to the entire scope of things that people were using computers for.

Brock: I see. Well, maybe unless there’s another aspect of your experience at Convair, maybe we could switch to the story of how you came to join RAND, in 1959 I think it was.

Boehm: Yes. Right. Again, I applied to UCLA and got accepted and went in—I really only interviewed at three places and they all were willing to give me a job. But again, with the previous experience that I had with RAND and that being in Santa Monica and the like, basically was—it was the deciding factor.

I had a few things that are non-technical here I guess. I didn’t have a security clearance when I went to—RAND was doing a lot of high security kind of things and they couldn’t let you have an office in the main building until you got your security clearance to come through. They had a place in the basement called “Pre-Clearance,” and they would give you something, an unclassified program, and send it out there and you would program away and then send them the code.

There were several people that were waiting for their clearances to come through. One of them was a woman who was normally a high school math teacher at Santa Monica High School. She wanted something different and got a one year’s assignment at RAND to program computers. So again, she didn’t have a clearance, and there we were down there in the basement. <laughs> We got to know each other, and she is now my wife.

Brock: <laughs> Thank goodness for Pre-Clearance.

Boehm: Yes, right. Once the clearance came through, then there were different teams in the computer center. There was an engineering team, which I was involved in. There was a physics team that was really doing more nuclear physics kinds of things. There was an economics team, and a business team. Basically, everybody was programming, but they were doing different kinds of it.

This was an interesting situation when RAND decided that it would upgrade its computing. The business people would want something that was good for databases and COBOL, and the engineering and physics people wanted something that was FORTRAN and a lot of computer cycles. We had a complex set of comparison criteria for all of candidate computers, and we had to negotiate the weights of these, say how important is COBOL and how important is FORTRAN. Everybody really had to interact with each other and negotiate with each other about what was going to be important about the computing facility. You got to understand what all those other people were doing.

Brock: Okay. What was the resolution of that?

Boehm: The resolution was an interesting counterexample. I ended up putting it in my *Software Engineering Economics* book. After all of these things got weights, so compilers had 24-percent of the weight and memory had 15-percent of the weight and computing power had 28-percent of the weight. Honeywell and GE and all of the seven dwarfs had reasonably competitive computing. And IBM really had two. What was best for us was at the top of the line at the time, Model 65 and they had just gotten the Model 67, which was doing virtual memory and the like.

Because it had some of these more advanced features, it scored higher in some of the areas. Basically, when we did the weighted sum of the ratings for the different computers, the thing that came out on top was the 360 Model 67. But once we started looking at it in more detail, we got these reports that it was spending most of its time talking to itself.

<laughter>

Boehm: Paging, and all of this kind of thing. And you were only going to get 15-percent of the computer cycles to do your FORTRAN or COBOL or whatever you wanted to do. We went back and looked at this set of ratings and weights and basically, yes, the weight for the computing power was 27-percent and there was only a weight—of that 27-percent, only 3-percent of it was how much of the computing power is being soaked up by overhead and in this case, it was 80-percent, but the weight was one-percent in terms of 27 times 3. Basically, we ended up getting a Model 65.

<laughter>

Brock: You re-weighted it.

Boehm: Yes, right and—well, what I put into *Software Engineering Economics* was that you really want to have a partly additive and partly multiplicative weighting function that said, yes, the delivered system capacity is the sum of all these weighted things that are giving you features. The amount of cycles that you're actually getting is a multiplier of that. The availability of the computer if it took you a week to fix it… So these multiplicative factors would make a more realistic decision criterion than just the pure weighted sum kind of thing for computing anyway.

Brock: Did that sort of computer purchase decision methodology, did that travel widely? Did people follow your footsteps, other organizations in doing that immediately, or is it something that you promoted?

Boehm: Not immediately. Again, this was in the '60s and the *Software Engineering Economics* chapter was published in 1981. After people saw that, and certainly when I was teaching software economics, and teaching it to a bunch of people that were local employees and things like that, they sort of picked it up and that sort of thing.

Osterweil: During my experience of the '60s, was that everybody bought IBM because it was considered a safe choice. People went through the exercise and weighting things and multiplying things and then the IBM salesmen somehow got the contract.

<laughter>

Osterweil: The top-level executives didn't want to take a chance on anything but IBM.

Boehm: Yes, that was the case at the time. It was interesting, when I got to TRW, TRW was big on faxes. But the business data processing people were all IBM and we, at one point, tried to bridge the gap and said, "Let's see what we can do in terms of an integrated support environment that fits both the engineering and the business data processing." We would ask the business data processing people, "What should we do about a forms management system?" And fundamentally, they would say, "Well, let us talk to our IBM people, then they'll tell us what we ought to do about that."

What we found was that they didn't have anything that was all that good. So we built our own and used it for configuration management, change control forms or travel forms or assessment forms or things like that. In 1982, it was the world's best forms management system. By 1986, it was still the best 1982 forms management system, but we were amortizing it as one company and then IBM and all the other companies were putting a lot more resources into building something that was better.

Osterweil: By that time, they realized that they were in the software business. So sometime in that period, in the late-‘60s, ‘70s, early-‘80s, IBM realized that there was money to be made in software and they started shifting their focus a bit. Although apparently, inside the company, it was quite a struggle.

Boehm: Yes, it definitely was. I can’t remember, were you on one of those committees with Fred Brooks because he was—

Osterweil: I’m sorry. What are those?

Boehm: Committees that said what should we be doing about this aspect of computing.

Osterweil: No, I wasn’t, but I was following what was going on.

Boehm: Yes. He had all sorts of stories about the hardware people versus the software people. Out came *The Mythical Man-Month* as a result.

Osterweil: Right.

Hsu: Was there a sense that IBM was somehow gaming these metrics, weighted metrics, to come out on top?

Boehm: Well, I don’t think they needed to because of what Lee was saying is that, yes, nobody’s going to fire you for choosing IBM.

Osterweil: Yes, mostly what they were, they were really successful as marketeers. This is back in the early-‘80s when I was a department chair, and the IBM salesman came to me and wanted to sell me—I think it was a 3330. It was old, old, completely obsolete machine. Wanted to sell it to me for a million dollars or something like that, and I threw him out of the office. He went down to see the dean and he offered to sell the dean three of them for three million dollars, and the dean came up to see me and he wanted to know why was I being so mean to the IBM guys.

I was in Boulder at the time, and IBM had a big plant out there in Boulder. I explained to him that it was an obsolete machine and they were taking advantage of us. So the dean threw the guy out of his office, and then he went to the university, and he offered the president of the university nine 3330s for nine billion dollars.

<laughter>

Osterweil: And the president of the university checked with the dean who checked with me and the president of the university threw the salesman out. And doggone it if he didn't go to the governor to complain about the president of the university. So that was one of the ways IBM exerted its marketing dominance. I remember telling an IBMer this story with outrage in my voice, and the IBM guy, he just turned to me and he looked and he smiled. He said, "I sure am glad those marketing people are doing their jobs."

<laughter>

Brock: I wanted to ask a few questions if I could about your experience at UCLA before talking about—I had some questions about RAND, but I know they're happening simultaneously, but maybe we could treat them separately and just… It was pursuing a PhD in mathematics at UCLA. If you could tell us about the nature of your work there, what you were working on, what your thesis was about. I was particularly interested to know if there was any connection to that National Bureau of Standards Western—oh, I can't remember the—

Hsu: Automatic.

Boehm: Yes, mm-hmm.

Brock: The operations that they had at UCLA.

Boehm: Yes. So, I knew a couple of the people that worked at that center and, yes, Walt [Walter] Karplus and Gerry [Gerald] Estrin, and people like that. But, yes, being a math major, I didn't really take any of their courses. I was interested in what they were doing. The thesis professor I had was somebody that I had worked with at General Dynamics in San Diego. He was hired to produce approximations for various functions that we needed to represent atmospheric density and pressure versus altitude or those kinds of things. So, he was very good at coming up with good, sometimes just polynomial approximations, but sometimes having sign or cosinefunctions in them or the like.

He got a job as a professor at UCLA and I thought, "Well, yes, I know him. He did practical work in approximation." So, he said, "Yes, I'll take you on as a PhD student." But he had gone much more into theory and fundamentally, he was not all that concerned about algorithms. I was hoping to find the next great new set of algorithms for certain kinds of functions and the like. He said, “Well, what I’m doing here is trying to generalize these things into, if it’s good for Euclidean spaces, is it good for non-Euclidean spaces or Hausdorff spaces.”

I labored away at trying to take the algorithms and abstract them into something that had much less strength and composability than the real number system does. And was able to show a few theorems that showed that, yes, some of the approximation algorithms would work in more generalized spaces, but nobody had any physical problems that fit those spaces. So, the title of my dissertation was, “Existence Convergence of Best Rational Tchebycheff Approximations.”

I was able to split it up into three little subsystems that were published in mathematical journals, and that was the end of my mathematical approximation career. Except that, yes, we still needed those things at RAND. Basically, when I got back, I was able to use things and then use other people’s algorithms and improve on them, but not really have any breakthroughs in approximation algorithms or anything like that.

Brock: Were you able to concentrate on this work or were you splitting your time between doing work at RAND and then work on your dissertation?

Osterweil: Those of us that have gotten PhDs in mathematics—mine was in combinatorics, which is a little more applicable, but I think back at those early days in the ‘70s when people like [Edsger Wybe] Dijkstra and Tony Hoare were, you know, spouting mathematic formulas. Those of who had PhDs in mathematics at least could understand what they were saying and had a better sense of what it was about, but what its limitations were that a lot of other people that sort of fell into computing and software development without a firm mathematical background, who I thought were much more easily sort of knocked over and blown out of the way. So I never regretted having that mathematical background because it allowed me to see what these other people were doing and what it was good for. And what it wasn’t good for.

Boehm: Yes. That’s definitely true, and I found the same thing in having conversations with Mr. Dijkstra and say, “Well, I know that testing only confirms the presence of errors, but not their absence, but that’s also doing proofs and you can make mistakes in proofs.” And he said, “Well, if you think hard enough, you can make a proof that doesn’t have errors in it.” Well, if you think hard enough, you can probably do that with software.

<laughter>

Boehm: One of the really highlights of interacting with that group of people was in 1978, I think it was. The Électricité de France company did a summer school every year. In some cases, it was on electronic transmission or other kinds of things. But in this case, it was software engineering. They invited three people. One was Tony Hoare. One was a French abstract person Jean-Raymond Abrial, I don't know if you remember him, Lee, and then the other was me.

Basically, each of us would have a lecture to do in the morning and a lecture to do in the afternoon and people got this wide sweep of formality-intensive or economics-intensive kinds of ways of looking at software. So, the other nice thing was that Tony Hoare and his wife were there with their kids that were aged 10, 12 and 14, and I was there with my wife who, our daughters were 11 and 13. We really sort of formed a family there for a while.

Hsu: Did you participate in the first Software Engineering conferences in 1968 and '69?[2]

Boehm: I was aware of them, but I didn't participate in them.

Hsu: Okay.

Boehm: Yes. One person from RAND did, Jim [James] Babcock, who actually left RAND and formed a company called Allen-Babcock [Computing] that did timesharing services and all that. But, yes, it was a landmark in the field.

Hsu: Yes. And earlier, we mentioned Fred Brooks and *The Mythical Man-Month* and in some of these analyses that you've done, you've talked about sort of the switch and the cost between hardware to software. This was a time where there was this talk about software crisis. What was your view of that at the time?

Boehm: Well, there were various aspects of the software crisis. There aren't enough programmers to go around kind of crises. Nobody could get their minds around these big complex programs and they're going to encounter situations which could devastate the company, country or collapse the stock market or things like that. I remember one famous debate that I think it was ICSE-8 [International Conference on Software Engineering] in London that Fred Brooks debated David Parnas on, can we really trust the software of ballistic missile defense? And, should we be investing in ballistic missile defense when we can't prove the correctness of the programs? Lee, were you in London for that one?

Osterweil: Oh, yes. I have a distinct recollection of that. In retrospect—well, no, at the time, I thought Parnas was a very foolish man to come out so strongly against what the government was pushing so

[2] NATO Software Engineering Conferences. Added by Frances Corry on August 27, 2018.

hard. In retrospect, I have more and more respect for him all the time because I think software people who understand the limitations of these things that we built really need to stand up and say something about their limitations.

And now with all this business about fake news and hacking and things like that, I think we need more people with David Parnas' kind of guts to stand up and make a fool of themselves, but to get people <audio cuts out> dangerous. So, I remember ICSE-8. I was not necessarily a big fan of David's at the time, but since then, I've become a bigger and bigger fan of what he did.

Boehm: Yes. Well, I was a fan ever since I think it was ICSE-3 in Atlanta that he gave his talk about designing software for ease of extension and contraction and saying—

Osterweil: And David, I'd like to throw in a story about Barry that he probably doesn't recall, but I think fits in very nicely. You talked about making the transition from a hardware-oriented world to a software-oriented world, where software is the big thing and there's a crisis and so on. This would have been around 1971, when I was a brand-new PhD and I had just started my academic career at the University of Colorado. And I was sitting in my office, I had just gotten done with my PhD in combinatorics and had a strong interest in programming and in software and so on and so forth, not sure what to do with my life and my career in a brand-new computer science department that had, I think, four faculty members at the time.

In the fall of 1971 was delivered to my office a copy of *Datamation* magazine, which at the time was a series, publication venue. And in *Datamation Magazine* was an article written by one Barry W. Boehm. And it was called, "CCIP-85," [Command and Control Information Processing] which was a study that he had done at RAND Corporation with some colleagues about what the shape of the computing world would be like in 1985, which was then 14 years in the future. And lo and behold, Barry had projected that by 1985, the current ratios of cost of hardware to cost of software would be completely turned upside down.

In 1971, hardware was 90-percent of the cost, and software was a 10-percent after thought. And Barry projected that in 1985, software would be 90-percent of the cost and hardware would be incidental, 10-percent. That was so heartening to some of us that we decided that this interest we have in software was probably a good thing to stick with. And it was absolutely formative for me. I've often reminded Barry this, in 1984, I asked him would he be publishing a "CCIP-99."

<laughter>

Osterweil: And that's what he did, he laughed just exactly like that.

Boehm: Well, yes. Basically, they didn't ask me in 1984, but, yes—

Osterweil: They should've.

Boehm: —I was at RAND and the Air Force wanted to have this study done. They asked an Air Force brigadier general, who was the best software person they had, and he said, "No, I've got another career thing I've got to do to make major general." They asked RAND did they have somebody who could run this, and they pointed to me.

We had a group of people that had representatives from the space community, the strategic air community, and the tactical air community, and a bunch of other people that were sort of contractor engineer analysts. We spent about a year going to the Strategic Command headquarters and then seeing what they were doing with command and control, and Sayan Mountain where they were doing the same thing for air defense, and Langley, Virginia, where they were doing a bunch of things for tactical air.

What we were finding in those areas—where originally people thought that the things that would be most needed in the mid-80s would be supercomputing and large screen displays—what we were finding was these people were fighting software all the way. The Strategic Air Command ended up with a command and control system that they had to rework 95-percent of to make useful. They thought they were doing the right thing. Basically, they subcontracted it to RCA [Corporation] in New Jersey, and they sent a dozen strategic Air Command people out to New Jersey to make sure that what the RCA people were programming was what the command and control system was actually supposed to do.

Two years later, they finished the project and then brought it back to Omaha, Nebraska, and basically said, "Oh, well, that's what we wanted two years ago. But the world has changed a whole lot in computing and strategic defense and offense and, you know, we can't use this." <laughs> There were software stories like that going on all over the place. Sayan Mountain had situations where their computers were all in the middle of a mountain and the humidity in these areas was—it was rotting the <laughs> computing equipment. But the worst things were really that they were ending up with the languages like JOVIAL [Jules' Own Version of the International Algebraic Language], that there weren't any—there are no JOVIAL programmers to carry on the tradition.

We found that these organizations were really fighting software, and not being able to do what they were hoping they could do in software. Part of it was requirements problems, part of it was reliability problems and then... We ended up that with this report that, again, as Lee said, there was each—in each of these places they were spending more on hardware and less than on—it was more on software on and less on hardware. And there was this curve that you could extrapolate it that it would end up going 90-10 rather than 10-90.

Again, you know the other thing that was nice was that yes, we had some companies that would do various studies for us, and then some of the companies had some data about what was happening to their productivity or what kind of things would make it more expensive to produce and the like. All of this hit the field at just about the right time, and *Datamation* was something that most of the people in the computing field felt was sort of the—it wasn't a professional journal, it was a commercial trade journal. But yes, somehow or another that article took off.

Osterweil: It really was remarkable. I think it really, if you traced it back it had an incredibly formative effect. I mean there was no software engineering discipline as a whole. There were no software engineering journals. There was some suspicion that software was of anything at all. I recall going to ICSE 2, which was the second International Conference on Software Engineering in San Francisco and I can't remember who it was. I think it was the guy from—anyway, I can't remember who it was. Then somebody asked him, you know, "What do you think of software development and software engineering?" And he frowned, and he said, "I think software is just mostly a form of low cunning."

<laughter>

Osterweil: It wasn't a serious thing. But then Barry and just a very small number of other pioneers grabbed a hold of it and pointed out that it not only was it a serious thing, but it was a serious problem. It really made a tremendous difference.

Boehm: Low cunning, it kind of like sounds like Morris [Maurice] Wilkes. I think I remember him saying something like—

Osterweil: Yes, I think you're right. I think it was Wilkes.

Boehm: <laughs> Yes.

Osterweil: Yes.

Boehm: Some other challenges that software had at the time was that it was invisible. One Air Force general that was commissioning software basically said, "You software guys are like the weavers in the story about 'The Emperor's New Clothes,' who said, yes, we are going to provide you with the most spectacular costume that kings have ever—emperors have ever worn, and all you have to do is give us some silver and gold and thread, and we will create this for you." So, he would come by every once in a while, and there they would be going, weaving and wafting and the like, but there wasn't anything in the middle of the weaving machine.

And he said, "Yes, it's coming along wonderfully. Just wait and you'll see how glorious it is to put it on." And so again, they declared victory and left the country as quickly as they could and the Emperor down the street with no clothes on. And everybody bowed except for a little four-year-old who said, <laughs> "The Emperor has no clothes." <laughs> This general said, "You software guys are sort of like those weavers—there's no tires to kick or anything like that. I go in and they say, ‘It's going fine, we're weaving away at this magic cloth.’" It was a hard thing to communicate with people about what it was all doing.

Hsu: Do you remember when the term software engineering started coming up? And also, what did it mean to you at the time?

Boehm: Well, I think the NATO books having that on their cover was really the main thing that got everybody saying that, "We know it's an aspiration rather than a mature discipline, but we need to get it to be more like physical engineering." Well, there were a bunch of other trends that were happening during the '60s that complicated things.

One was really that we ran out of engineers to turn into computer software engineers, and so people had to hire music majors and art majors and philosophy majors and the like. You got, in a lot of cases, people that didn't really want to act like engineers, and would end up creating interesting stuff, but it wouldn't do the job and it wouldn't interact with anybody else's software and the like.

I think that I did this one paper that Lee actually commissioned me to do as a keynote that ICSE 2006 in Shanghai, that basically said, "Is there a way we can explain the evolution of emphases in software?" What I found was that there is this Hegelian approach to philosophy—

<laughter>

Boehm: —that says basically somebody presents a thesis and then somebody else presents an anti-thesis and then somebody converges that into a synthesis and that becomes a thesis and somebody comes up with an anti-thesis. In the '50s, the thesis was, “You should engineer software like you engineer hardware.” And then in the '60s, with all the art music majors and the like, they said, "No, it's a craft. It's not an engineering discipline. You need to be inspired and to do it well," and the like. And the nature of the thing said, "No, you got to turn it around and make it more engineering-like." There was this sort of thesis, anti-thesis synthesis kind of thing that was going on across the decades. There was a Dagstuhl a week on the history of software engineering and we were trying to figure out can we explain <laughs> why it went the way it did.

Hsu: Can I ask one more?

Brock: Please.

Boehm: Yes.

Hsu: Earlier you mentioned Dijkstra.

Boehm: Mm-hmm.

Hsu: When did you first interact with him or meet with him? And what did you think of, you know, the things that he was advocating, structural programming and program correctness? The usefulness of those approaches.

Boehm: Yes. As part of the follow-up to the *Datamation*—the study that led to the *Datamation* article, I did try to help the Air Force determine what should be in their research program. One of the things that was really important was trying to make it more reliable. I went to the Netherlands then and had an afternoon with Edsger Dijkstra. I think what he had was really a lot of really good ideas, but he had sort of a limited sphere of experience. I would ask him a question and there would be silence for about two minutes and out would come this two-minute lecture. <laughs> Wonderfully organized, thoroughly thought through, and gave you more than you thought you could get as an answer. But it was sort of an interesting afternoon. It was kind of…

<laughter>

Boehm: I found that most of his examples were a fairly formal kind of examples of what people did for programming, and that if you were trying to develop programs to help educate people or help people negotiate things or the like, there were a lot of human unexplainables that trying to formalize the requirements, and address building a computer program that satisfied the requirements, was not the solution. A lot of times it was prototyping this and prototyping that and saying, "Is this going to satisfy people?"

At one point, he did one of his favorite EWD notes and said, "Yes, I have just reviewed this paper on human computer interaction and you can't formalize it. And if you can't formalize it, it has no business being in software engineering."

<laughter>

Boehm: Again, that's one definition of software engineering. Unfortunately, some people really sort of—well, he himself basically said, "Just getting the program right is as much as my brain can do, so I don't think software people should really think about the requirements. I think they should let somebody else give them the requirements and then you can get your head around that and produce what they asked for." But again, yes—

Osterweil: So that wasn't in NATO meetings either. I've read the proceedings pretty carefully because I think I did a panel session on the NATO Conference. I'm not sure whether you have the same recollection or the same reconstruction of history as this. But my sense is that the 1968 NATO Conference basically pitted the United States against Europeans, and there was a deep philosophical rift. The Europeans were strongly influenced by Dijkstra and by Tony Hoare people like that who felt that this was serious mathematics and it was heavy duty stuff, and all of the things you just said.

But the Americans came impatient and, "We've got to write a program. We've got to fly missile planes and we got to get stuff done." And in 1969, when they got back together again for the second meeting in Rome, my understanding is that it was pretty ugly, that there was abuse on both sides and in fact, there wasn't another International Software Engineering <audio cuts out> for at least another four years. I think you and I were both at that meeting in New York in 1973. I had no idea what was going on. I just knew that there were a lot of people that were very mad at each other.

<laughter>

Osterweil: And, you know, I can almost trace to this day the fact that there is a European view of software engineering that's much more mathematical and much more formal, and then there is an American view that seems to be much more pragmatic. I think the difference is much harder to detect right now, but certainly in France there is hardly any practical software engineering education that goes on. It's all mathematical and very formal.

Boehm: Yes, I think that's definitely the case. Although there are exceptions. The Scandinavians I think are much more people-oriented, you know?

Osterweil: Yes. You have that recollection in the early days after the software engineering, the NATO meetings that there was an American side and a European side and they really didn't seem to see eye to eye at all.

Boehm: Yes, there were some exceptions, that again, my experiences with Tony Hoare and the French summer school and things like that got me to appreciate what he could contribute and I think got him to appreciate what American perspectives could contribute. Again, I think he has been one of the more practical people. Certainly, when he left Oxford and went to Microsoft and then and said, "Yes, let's see what we can do to make things more rigorous than they have been."

Osterweil: Yes.

Brock: Well, one quick question about your Air Force study, circa 1970. It seems like the problem or the causes for the cost dominance of software that you were getting at, to me, strike me hearing you describe them as more about more social problems or organizational problems than something that is fundamentally a technical challenge. You know, it's not like solving Fermat's Last Theorem. It's not like making fusion energy. It's more the dynamics of a large process involving lots of organizations and changing conditions on the ground. Is that a fair characterization?

Boehm: That's a fair characterization of some aspects of the situation. I think one of the other economic trends that really pushed people more towards software was really when you could do electronic upgrades of software. In 1960, an Air Force plane, about 10 percent of the functionality was done in software. But a lot of the other things were done by pneumatics or hydrodynamics or mechanical linkages and things like that. If you wanted to or needed to upgrade your system, fundamentally you had to get each individual plane and go off and fix it.

People found that there were ways to build these airplane buses that you could send electronic signals around. For a lot of the things that you wanted to change the functionality it was easy to just change the software once and then send it off to upgrade all the fleet. By 2000, 80 percent of the functionality in an airplane was being done by software. One general said, "About the only thing we can do with an F-16 without software, is to take a picture of it."

<laughter>

Boehm: Again yes, I think it's going to over 90 percent for the F-22 and the F-35.

Brock: Yes, there it just seems there's so—

Boehm: Yes.

Brock: —most of what you read about that F-35 aircraft is about software.

Boehm: Mm-hmm. Yes. Right.

Brock: Well, if I could, I'd like to direct us back to some developments during your tenure at RAND that I thought were very interesting that I think that you were involved with. Before asking you about the well, one that I know that you were involved with, the rocket simulation program, I did want to ask you about the RAND Tablet and that whole development, which Hansen and I have recently learned about from Alan Kay, who—

Boehm: Oh good.

Brock: —sings its praises dramatically.

Boehm: Yes. Uh-huh.

Brock: I wondered if you could tell us a little bit about your experience of the RAND Tablet effort?

Boehm: Sure. Yes. What it enabled you to do is to write on a tablet, and if you had a good character recognition piece of software you could get it so that people could work it pretty well. At the time RAND was looking for ways to apply the tablet to help get various kinds of jobs done. We had this ROCKET [RAND's Omnibus Calculator of the Kinematics of Earth Trajectories] program, and fundamentally it was something where you put a bunch of inputs in and a bunch of outputs out and then you got a bunch of stacks of paper that that showed you what the trajectory was like and then how accurate it—and things like that.

Between Tom Ellis, who was the guy who really invented the way the Tablet worked, and Gabriel Groner,, who built the character recognition system, and Keith Uncapher, who was the head of the part of RAND that was doing all the stuff for DARPA [Defense Advanced Research Projects Agency] said, "Yes, why don't we build a graphic version of the ROCKET Program? We will put the input forms on the screen and people can write the takeoff weight and the atmospheric model that they wanted and the thrust of the fuel flow model they wanted," and the like.

The system would chunk away, and in front your eyes there was trajectory, the rocket going the way you wanted or in some cases the ROCKET program would allow you to do tradeoff studies then and say, "Suppose that we had—let's try different engines and then see which one does the best job of meeting the range criteria." You'd get four things going along, and then you could see which ones got the farthest and the like. This was something that the RAND rocket engineers liked a lot.

Basically, what we ended up doing and then trying to publicize it was doing some UCLA short courses on graphical methods of rocket engineering. Surprisingly—well, there were a couple of places that were doing similar things, that Lockheed had a group in Georgia that had built something that was optimized around Lockheed products, but it could give you graphs of performance and—

Brock: On a screen?

Boehm: Yes.

Brock: Mm-hmm.

Boehm: But fundamentally, yes, the RAND engineers were doing their work on a 360 Model 50 that DARPA had paid for, so it was free for them.

<laughter>

Boehm: But if you wanted to hire an hour of 360 Model 50 time, it would cost you $50. Outside of RAND, it didn't really have a big market. The thing that was interesting was that we had done it in a way that you had different modules for entering the inputs and displaying the outputs and doing various kinds of comparisons and the like. There were other people at RAND that had similar problems. There were people that were doing complex medical analyses and then saying, "Well, yes, what we want to do is to see if we are feeding this person this kind of medication at this rate and this other kind at this other rate, is it going to lead to something that makes them more healthy or less healthy?"

It was the same basic problem, that you wanted to put in the rates here with your little handwriting and then see the curves with... We did find that we had created a GUI [graphical user interface] builder. And we didn't realize that GUI builders were anything that people worried about <laughs> them until a few years later. Again, it was a non-economic GUI builder for anybody outside of RAND but it served a pretty good purpose. We called it Programmer-Oriented Graphic Operation or POGO.

Brock: Oh, I had that on the question list. I hadn't realized that it was—

Boehm: And we wanted to put a Walt Kelly Pogo, and we got this nasty letter from his publisher.

<laughter>

Boehm: "You can't do this."

Brock: So this whole effort around the RAND Tablet and its various uses, did this run across the '60s or—because POGO was '67, is that right? Somewhere around there?

Boehm: That's about it, '67, '68.

Brock: Yes.

Boehm: Yes. Mm-hmm.

Brock: So it must've been running for several years before—it's the mid-60s was this effort, would you say?

Boehm: Yes. Uh-huh.

Brock: Okay.

Boehm: Yes. Uh-huh.

Brock: And when does the whole activity start up at [University of] Utah in computer graphics—around the same time, is it not?

Osterweil: The late '60s? Yes.

Boehm: It was around the same—yes, it was basically when Ivan Sutherland left DARPA and went to Utah.

Brock: Right.

Boehm: Because again, he was sponsoring that kind of stuff when he was at DARPA and he was doing that stuff at MIT before he went to DARPA to sort of evangelize doing more and more of this.

Brock: Oh, right.

Boehm: Then getting DARPA to sponsor more and more computer graphics technology and—

Brock: So was Ivan Sutherland sponsoring the RAND Tablet?

Boehm: Yes. Uh-huh. Uh-huh.

Brock: I see. I hadn't made that connection before.

Boehm: Yes. Yes. Uh-huh. Yes.

Brock: Okay. That makes perfect sense. <laughs>

Boehm: Yes. Right.

Brock: Another thing I wanted to ask you about RAND was this—if it is also a fair characterization, that from the outside it seemed like it had an interesting mix of being closed and open. In that there's certain activities that were very academic, if you will, publishing, presentations, mixed with work that by its nature had to be closed, for military work or government work. I don't know, I think that's an interesting mixture. I wondered what you thought about that.

Boehm: Well, it sort of reached a—almost a crisis point when RAND decided to create a Domestic Studies division. And the Air Force said, "You're the Air Force's think tank. We don't want all your smart people to go off and solve New York City's housing problems and firefighting problems and things like that. It's going to be too hard to separate the organization into an unclassified bunch of people and a classified bunch of people." And so it did. There were parts of RAND that were doing Domestic Studies and they couldn't penetrate the other part of RAND. People would get together for lunch and <laughs> things like that, but it was a problem.

Also at about the same time Daniel Ellsberg decided to release the Pentagon Papers and RAND was really in the doghouse as far as supporting the Defense Department after that. It could've been a existential moment for RAND that it might not survive all of this. <laughs> But eventually RAND did a good job in getting a bunch of really influential people on its board of trustees that would, if the Air Force generals were getting too feisty about something, they could go to the Secretary of Defense and say, "Can we talk to these people for a bit and then see if there's some more constructive way we can move along?" And that sort of thing. I think those kind of things helped the Air Force get what it wanted and then the Domestic Studies division people got what they wanted.

Brock: Were there tensions within RAND about the work being done for maybe the military more broadly during Vietnam? What was the experience of like that from inside the organization?

Boehm: I think for most people, Vietnam was not at the center of their attention. It was mostly the people that got—like Ellsberg and Amrom Katz, who went over and said, "How can RAND wisdom conquer the Vietcong?" And then finding that they really had no solutions.

<laughter>

Boehm: Then saying, "This whole war makes no sense." So yes, so I yes—it didn't really, the average RAND person didn't really worry much about it, except for the fact the company might <laughs> go out of business. There were other people that were uncomfortable about RAND analyses. Herman Kahn did these things about the “megadeath.”

Brock: Right.

Boehm: And, “How can you talk about people that way?” It just, “I know you're just trying to be engineering and scientific and algorithmic about different kinds of ballistic missile offenses and defenses and its casualties and the like, but this is inhuman.” So yes, a bunch of people didn't associate with other people. Richard Bellman was a good example. He invented dynamic programming but he was a pure mathematician. He tried to apply dynamic programming to all sorts of medical problems and things like that, and said , "I much try to make people healthy rather than kill them."

Brock: Right.

Boehm: "And I don't really want to talk to this guy." There were things like that that were going on inside the company.

Brock: Where did you find yourself?

Boehm: I must confess that in a couple of the system analyses that I did in trying to optimize where would you want to put your satellites and things like that, that I never counted casualties or anything like that, but basically said, "Here are the places that if you wanted to drop something that had an electric magnetic pulse that shut down all of the military facilities. Here's the way you would configure your satellites and the like." But that's about as close to that kind of stuff as I got.

Brock: I did some oral history interviews with Jay Last, who was one of the people responsible for the silicon integrated circuit.

Boehm: Uh-huh.

Brock: Which was, of course, part and parcel of this missile work ICBM [intercontinental ballistic missile] work. In those interviews, he made a point to say, "You also have to understand that it was a scary time on the world scene." So that's not to be forgotten as a context.

Boehm: Yes, and I got involved in the ARPANET [Advanced Research Projects Agency Network] basically because of those kind of things. The Minuteman Missiles had various kinds of communications support things that were basically wires, and so there were concerns that if either airplanes or missiles came along and cut the wires between these things, that they couldn't receive their messages to fire, or things like that. The packet-switched network approach looked like it might be a good solution for that, and I got involved in analyzing those kind of things.

Coincidentally, it was my wife who was the key contributor to this, and Paul Baran was the engineer who was doing this series of 14 reports that said, "Yes, here's how you can do packet-switched networking and here's how you can reassemble the messages at the other end, and here's how you can transmit things, and here's how you could worry about overloads and the like." She was programming during this year, and Paul Baran said, "I need somebody to simulate these and make sure that this kind of routing is going to succeed." So she produced something, and basically showed that it could address all of the potential problems using the kind of algorithms or some variation of some algorithm. Paul was a nice guy and he named her the first author of that report. She really has a minor place in the history of the ARPANET and the Internet.

Brock: What is her name? I'm sorry, I don't know.

Boehm: Sharla. I'll give you a website. There's a guy who calls himself “the software curmudgeon” or something like that, and he was surfing the internet and he found this set of Paul Baran things and found this one by this woman that did this analysis and was the lead author of it, and put this blog out that says "Who in the heck is Sharla B. Boehm?"

<laughter>

Eventually I got wind of it and looked at it, and responded to him and said, "Yes, that's what she did, and here's some more about her. He added to his blog, and so it's something that if the Computer History Museum is interested in women in computing—

Brock: Absolutely.

Boehm: —this would be something that you might be interested in looking at.

Brock: Very much so. So she did a simulation of a packet-switched network along the lines that Baran was proposing and saw that it had the…

Boehm: Yes, that you could make it work, yes.

Brock: Wow. And did she, for her professional career, was she at RAND for her whole run? I guess I could talk to her. <laughs>

Boehm: No, well, she was going to go to RAND in 1958 and she had also applied to teach Air Force dependents in England and decided, well, I'll spend a year and live in London and teach Air Force dependents. But they gave her another year off from the high school, and so she did a year at RAND, and then I think in summers she would go back and do some programming.

Brock: Was it in one of these summers that she did the network?

Boehm: I think, yes, it was more in the early ‘60s rather than late ‘50s.

Brock: Right.

Boehm: And then we had our first daughter in 1965 and she decided that she'd stop teaching, she'd stop programming, <laughs> be a girl scout leader and things like that.

Brock: Thank you. Well, I would love to see that. Please.

Hsu: I have a question about System Development Corporation and the work that they did on SAGE rapidly expanded the number of software programmers. Did you have any relationship to that?

Boehm: Some. Again, I interviewed for them and they offered me a job and I got an idea of what SAGE was all about. And then, fundamentally, I stumbled across a couple of papers that one Herb [Herbert] Benington had done in 1956 that basically said, "Here's the waterfall model," 14 years before Winston Royce wrote the waterfall model paper. “We succeeded at this because we were all engineers.” Again, this was sort of a initial thesis that says, "If you want to get good software, get engineers to produce it."

Then there was another paper by a guy named Bill [William] Hosier who went through a bunch of difficulties that they had had, and what kind of lessons learned that there were from these. What I did in 1979 was to—Lee, this was the year that you did your “software processes” or “software too” things. I had a session that was sort of, let’s look back at the history and see what Herb Benington and Winston Royce and—one of the other System Development Corporation people, Pat Haverty, did the Hosier article because he wasn't available then.

Osterweil: '86 or 87, yes.

Boehm: Yes. Basically all of the lessons learned in this were still lessons learned in the late ‘80s. What I was trying to do with this session and this article was to say, yes, these are good things, but that was published in 1961. <laughs> How come we've gone another 15 years and still haven't learned the lesson?

Osterweil: And it gets republished every couple of more years because the lesson still hasn't quite been learned and the solution hasn't been found. But, I mean, from some point of view it suggests that software engineering has got some deep and enduring problems that have not quite actually been crystallized yet. But they are deep and enduring because people keep trying and not quite figuring out how to do these things right.

Boehm: Yes. One of the things that we are currently doing at USC [University of Southern California] is building these technical analyzers that fundamentally look for bad smells in your software and dead code and duplicate code that's, if you want to modify it, you'd better remember where you put the other copies and things like that. These are becoming really valuable. We've been doing a deep dive on technical debt and software maintainability. One of the things that we found was that many of the main sources of technical debt are non-technical. We have a top ten list of non-technical sources of technical debt, like the conspiracy of optimism, basically.

<laughter>

Boehm: Where the people that are wanting to get their program funded from the government and the Air Force or the Navy or whatever—Interior department or whatever, they tend to be optimistic about what they can do for a given budget. They know they've gotten themselves into a tight situation, so a lot of government agencies still use the successful bidder criterion of, "We are going to award this project to the lowest cost, technically acceptable bidder.” If you do a cost estimate, you say this is going to cost $10 million, and you know that if you bid $10 million, somebody's going to underbid you, so you try to be optimistic about saying, "Well, if we get the A-team for this and all the COTS [commercial off-the-shelf] products interoperate and all those kinds of things, we can get it done for $7 million."

And then we have this thing called the “cone of uncertainty” that basically says, before you build it you really don't know exactly how these things are going to do, and it could cost you 10, it could cost you 20, it could cost you five. The only way you're going to find out exactly what it’s going to cost is at the end.

What people tend to do in the conspiracy of optimism is to choose the lower thing of the cone of uncertainty and get the bad news as they go along.

<laughter>

Boehm: A good example is the F-22. When they started, they were going to build 750 airplanes for 26 billion dollars, and now what they have is 187 airplanes, or a quarter as many airplanes, for 78 billion dollars or three times the price. It's a factor of 12 that happened there. If your software is going to end up in situations like that, well, the first thing that hits you is really the system engineers don't have an adequate budget, so they're going to do a poor job of system engineering. All of that is going to become what one of our other professors calls “architectural debt,” and all that translates into software debt. But it's a non-technical source of technical debt.

Brock: Why the choice of the term “debt” for that?

Boehm: This was something that [Ward] Cunningham, when he was doing agile methods, found that in order to get something done quickly, it was worth deferring certain testing, and careful buddy-checking, or inspections, or things like that.

Brock: Oh, I see.

Boehm: In those cases, it may be for good reason, because you want to be first to market or you want to get the devices out that are killing soldiers with improvised explosive devices and things like that. Basically you say, "I'm willing to postpone fixing these things,” but what Ward Cunningham found was the situation was that the more he put it off, the more expensive it became to fix and that it was acquiring interest on the debt.

Brock: Oh, interesting.

Boehm: So that's where the analogy came from.

Brock: I see now.

Hsu: Is this a good time to talk about sort of the history of these sorts of engineering processes? Waterfall and spiral model and agile, and go over that?

Brock: Sure. I mean, Barry would know better. <laughs>

Boehm: Yes, yes. One of the things that I really enjoyed was working with Winston Royce. Everybody looks at that and says—what he was trying to get everybody to do was to do a one-pass through and, “Don't think about the design until you finish the requirements because you'll be contaminating the requirements by pre-deciding the design.” And, “Don't do the design, don't do any coding until you finish the design.”

For the kind of things that TRW was doing up here in Sunnyvale, was keeping track of satellites. The satellites are very deterministic, and all the equations are well-understood. So you can do requirements, then design, then code and then tests, and it works pretty well. But what Win and the rest of TRW were finding was that as projects became more human-intensive, that you had to get the user interfaces right and things like that, and that you couldn't really write down what should you put on the screen to make sure that the people make the right decision about the situation.

So we, at TRW, started investing a lot in rapid prototyping technology and the like, and this isn’t really waterfall. What, in various conversations, we came up with was what was really happening was that there was a lot of uncertainty going on, and that what prototyping was doing was an aspect of statistical decision theory that said, "You're buying information to reduce risk." That's fundamentally what got the spiral model going was to say—and what Win Royce really said in that 1970 article was not that there's just one path. He did say, "You ought to be doing prototyping,” and there's this secondary thing that says, "Build a piece of it and see if it's going in the right direction, and then go back and do the waterfall."

What the spiral model was doing was trying to formalize that, and say that what you want to do is look for the uncertainties that you're not sure about. Uncertainties are probabilities of loss and probability of loss times size of loss equals risk exposure. The big risk things are the things that you ought to be buying information to reduce risk. Sometimes they're user-interface things. Sometimes they're scalable algorithms and the like, but sometimes they're scalability of COTS products or interoperability of COTS products. Basically before you embark on some risky requirements, you should buy some information and see whether that's the right way to go or not. That's what the spiral model was trying to do.

Unfortunately, the original spiral model paper tried to compress all of these things into a single diagram. It was easy to misinterpret the diagram as saying that what you're doing is exploring and identifying the needs and opportunities and risks and resolving the risks and then going around in the spiral. So yes, if you want some entertainment, just Google on “samples of spiral model diagrams,” and you'll find the craziest bunch of things. <laughs>

We went through various iterations of the spiral model. We called it the “win-win spiral model” for a while because we were also trying to negotiate win-win conditions among the various stakeholders in the project. But we eventually converged on the “incremental commitment spiral model” which is the title of the latest book that defines what it is. That tries to address what happened, after rapid prototyping and buying information to reduce risk, was that the agile methods kinds of things started coming up.

Again, what we try to do in teaching software engineering at USC is to have people get together in six-person teams, and meet with a client that they've never seen before, and build them a piece of useful software. Fundamentally, they therefore know where the requirements come from. They do an initial get-together in a room that is orchestrated by a professor in the business school is just super at getting people to quickly understand each other. Then she calls on them and says, "Tell me what you did about learning what the problem was and learning what the solution was." Then two days after that, they have to go and visit their clients' facility and see what are they doing, and what do they want to do better, and who are the people that are involved and the like. Basically we apply the spiral model, and we're doing all of those courses and de-bug it as we go along.

One of the things that we found that was a problem was that our documentation got super-heavy, and that we really ought to, if we're going to teach people how to do 21st century software, we should be teaching them agile methods. Fortunately at the time I was writing a column for *IEEE Software* and got connected with some of the agile people to write some pieces and say what were they trying to do, and then really learned a lot from what they were trying to do. It's something we should be teaching our students much more about, but it doesn't cover everything.

There are other aspects that we found, like trying to converge on people understanding each other's problems and solutions before you go off and lay the first line of code. We have a negotiation groupware tool called the—it's gone through six iterations. It was called “Easy WinWin” for a while and now it's called “Winbook” because it looks like Facebook and the clients can use it more easily. Basically, what the developers and clients and clients' maintainers do is to say, as an architect, “I want the system to be able to address scalability up to the kind of things that you need.” And the clients say, "As a client, I run a community service organization and I need to keep track of my events and my volunteers and my finances and my donors and things like that.”

Basically, what they do is find things that if they agree with them, then it becomes a requirement. If they disagree, it becomes an issue, and they need to sort of come and do some kind of horse trading that gets them to something that is a win for both of them, that they're better off than they would have been otherwise. So nobody gets everything they want, but everybody gets something better than what they have. Again, all of that is really saying that just getting a bunch of people around and doing a 30-day sprint and then saying, "Is that what you want? What do you want next?" It sometimes works but sometimes gets people, as Ward Cunningham found, in increasing technical debt because they will go off in a different direction in the next sprint, and getting the things to complement each other isn’t as easy as it might look.

Hsu: What other sort of strengths versus weaknesses between waterfall and spiral and agile, these different models, do you think there are? How can we improve on these processes?

Boehm: Yes, so what we found was that, yes, the latest version of the incremental commitment spiral model basically says, what you're going to do is elaborate the definition of the system one more time and determine and show evidence that if you build it to the architecture that you're proposing, here's the evidence that it will be buildable in the budgets and the schedule and the plan, and satisfy the requirements and satisfy all of the stakeholders' needs.

If the evidence is insufficient, basically you have an uncertainty or a probability of loss. Again, multiplying that by size of loss gives you risk. You basically say, as a stakeholder, “Do you accept this level of risk?” If everybody agrees, then you go forward. In some cases, again, you may say, "I need to get to market quickly and I'm willing to leave the risks there and then build up some technical debt."

There's really, out of each one of these decision points, that you could either go forward into the next phase or you can say, "This is a medical product that needs to be safe and we haven't done enough safety analysis, and we should do that before we commit to this particular architecture." Or, you can find that you thought you were going to be first to market, but there's already three companies that have 80 percent of the market share, so you want to have an off-ramp that says that it's better off to drop this project and save your resources for a more competitive project.

Or, you can have a situation in which your evidence is that you're building a small system for somebody to keep track of their donors and the like, and there's an enterprise resource planning system that their company has a subscription to, and they've built more complex things than this using the ERP system. So you really don't need to go into another phase that is going to do a business case, because you already own the thing and it has done bigger projects and you don't need to do an architecture analysis because the architecture supported a more complex thing. In those kind of cases, you can say, "We don't need anymore upfront stuff. We can just go do sprint one and we know there's low risk of this running into problems.” That sort of thing.

What we've done, basically there are three definition phases and four ways out of each one, so there's four cubed or 64 different ways that you can go through this, and that's too many. What we've done is to identify some common cases and say, "Here are half a dozen decision criteria that say, how big is this system, and how complex is it, how critical is it? Is it safety critical or is it just something that's going to give you a loss of comfort if your video game crashes or something like that? How COTS-intensive is the system? How capable are the people in being able to do this kind of thing?”

There's one more criterion, but basically, how volatile are the requirements? If you have a situation where the worst requirements volatility things are in mergers and acquisitions where people try to merge their corporate infrastructure software, and every week they find some new problem that says, "We can't write the requirements for this thing, and will be rewriting requirements all our life rather than building something that gets closer and closer to what we need."

What we found is that, yes, for various settings of, if the system is small and it's low to medium criticality and your people are agile-experienced and the requirements are volatile, you want to do pure agile. Just don't try to write any requirements. Just do a piece and then get back to the clients and say, "Well, what are the top priority things in your backlog and let’s do those for the next piece." If it is something that is a little bit bigger and a little bit more critical, we say—and a bunch of companies have done this very effectively—there is something called “architected agile” that basically says before you do a programming sprint, you do an architecter sprint, and you make sure that fundamentally what you choose in terms of architectural structure, and choices of COTS products or cloud services and things like that, are going to be satisfactory and scalable enough to satisfy this system, and then you can go agile from there.

There's about a dozen of those special cases. If you're migrating from some horrible combinational legacy software, then there's a special kind of thing, or different kind of process that you do for that. If you're doing something that's security-critical, and then you need to use formal methods. Again, it's trying to come up with something that is not perfect but gets you pretty close to where you want to be in terms of what you want to do next and the like.

Hsu: Just one more?

Brock: Sure.

Hsu: Another thing that has been I guess more in the past, but sort of there was one strategy for improving software engineering, was to build certain techniques into the tools or into programming languages like structured programming or object-oriented programming. How have these attempts, sort of technical solutions, actually worked? You know, to improve things, versus how much have they fallen short in solving the problems?

Boehm: Well, I think what happens is that, yes, if you have a mature domain, and one that's not too complex, that's what all these enterprise resource planning kind of things will do for you. They say that there is a pretty safe architecture for doing these kind of things. There are ways that SAP or Oracle or the like can be, will give you some tailoring languages so you can tailor the things to what your processes are, and you can tailor the reports to what your reports need to be and the like. Those are really a sort of a spectacular win for that kind of thing.

But there are still problems that are beyond our ability to really come up with packaged solutions or something other than just feeling your way. I think the biggest examples are crisis management systems.

We've got to do better than Katrina or Puerto Rico or the like. In those cases, you really have a lot of people who come out of different cultures. They have made different choices of architectures and data structures and things like that, and they have different—the police are trying to reduce crime and the firefighters are trying to get the fires out, and then the hospitals are trying to get the people healthy again. Even within those areas, they have incompatible data management systems and communication systems and the like. Those, I think, really sort of the challenges for the future, and how can you build one of these? In *The Incremental Commitment Spiral Model* book, we have an example of a medical first responder system that tries to say, “Here's one way that you could get through these,” but it doesn't really, in a 250-page book, solve the crisis management system architecture problem or anything like that.

Brock: Well, I just wanted to, in the time that we have remaining with you, I did want to talk about your tenure at TRW and your overall experience there, and then make sure to have a chance to talk about your service with DARPA and the move to USC. To start that, maybe we would talk about your, you know, what led you to shift from RAND to TRW in the early '70s.

Boehm: Well, it was basically that CCIP-85 study, said that the software is going to be a real complex problem to try to get your arms around, and try to quote "solve," and the biggest software project at RAND had four programmers on it. Basically, what I felt I needed to do was to go to a place that did a whole lot of software, and so I sort of reengaged with System Development Corporation and what Thompson Ramo Wooldridge had become, which was TRW. Fortunately, TRW was saying, "We really feel that we need to get more up to speed in our software technology." This came right after the *Datamation* article, which basically gave me a little bit of credibility. They said, "Yes, we would like you to be our director of software research and technology."

Fundamentally, what this involved was taking an inventory of what they had done, which was a lot of really good stuff. They had a bunch of tools that they had built that helped you organize testing better and count the number of branches that had been exercised in a test and the number of instructions that had been. If you could get to 100 percent, then you felt that things were pretty well-tested. We had gone from manual standards checking to having some automated standards checkers, and some other things that we had gotten from other places that had good approaches to doing requirements. Dan Teichroew had something called PSL/PSA, problem statement language and analyzer, that we got a lot of value out of.

Since I had had all of this opportunity at the courtesy of the Air Force to go around and see what everybody was doing, I can bring some stuff in and capitalize on the stuff that was there, and respond to needs that the company said that they needed. For example, we had capitalized on the Win Royce waterfall article and then said, "Yes, we are your software developer of choice because we are the masters of the waterfall model."

What happened was that we were using it in Sunnyvale and a couple of other places, but there were other customers who said, "You're not doing that on our project. Why aren't you doing this on our project?” So our division general manager said, "We need to have a set of TRW policies and standards and say, ‘This is the waterfall model and this is how you do software at TRW.’” We went through a process over a year that converged on about 20 policies and 300 pages of standards and the like. Went through an education process where we first educated all the department heads to see what it was all about, and then have them teach all the people in the department about it, and follow that up with a 40-question analog to the California driver's test that says, "True/false.”

<laughter>

Boehm: “If you don't get 35 of these right, you can't drive a software process from the TRW." What we had unfortunately done was brainwashed people into using the waterfall model. Then when we started getting these user-intensive decision support systems, you really didn't want the waterfall. You wanted the prototyping and things like that. It was a real challenge to try to change the culture back again. Basically, people said, "You can’t do the prototyping on that project because that’s programming before the critical design review, and we can't do that. It's illegal and immoral and unethical." <laughs>

It took a while, but we did finally get the best of both worlds, and then re-did the policies and standards really around a more risk-driven approach. So that helped a lot. I found that my job would take big changes like that every once in a while. One of the next ones was the division general manager called me into his office and said, "Over the past three weeks, I've had to sign off on projects that committed us to build the software for 50 million dollars each, and nobody could tell me why it wasn't 25 or 75 million dollars. We've got to be able to do better than this." That was really sort of the beginning of the COCOMO [Constructive Cost Model] cost model.

Since he was the division general manager, he could lean on people to give us data to calibrate a model, and he could lean on parts of the company to give us smart people to converge on what the cost drivers were for the model. Finally, we got 20 projects and found that with the data that we had, we could calibrate them to the parametric model that the team of experts put together. That was a really good thing for the company. Unfortunately, we would send in a proposal and then say, "We have this much higher level of assurance than other people do about the costs that were bidding because we have estimated it with the COCOMO model, which is TRW-proprietary. And the customers would say, "Well, we need something that we can compare your competitors with.”

Brock: Oh.

Boehm: “And if you won't give them the model, we can't really just count on your use of it.” It took a while, but the people that wanted to keep it proprietary basically slowly said, "Yes, it's better if we make a public version of it." What I found was that we still had a few more TRW projects that we could add to it, but I was teaching these software economics courses at UCLA. The students were from companies, and I would say, "One of your assignments is to provide me with at least one project that uses this model and calibrate it to the data that you have on how much effort it took."

What came out of that was that the business data processing people had really different, you know, the cost drivers were very similar but the exponents were—these economies of scale were much higher for the high-tech government kind of things than they were the business data processing things. We had to generalize the model in some way to cover these other kind of approaches.

Then TRW said we will let you publish this in a—well, another thing that happened was around the mid-70s, people were starting to write software engineering textbooks. I said, "I want to write a software engineering textbook." I got TRW to say I could take a semester's leave of absence, and in this case I went to USC rather than UCLA because Ellis Horowitz, who was there, said “We need something like this for our program.” Basically, I taught a course using the kind of things that I was going to be putting in the textbook, and busily sat there in my office trying to write the pages in the textbook. The first six weeks were the most productive in my life because it took them six weeks to put a telephone in my office.

<laughter>

Boehm: At the end of the semester, I had about 300 pages of stuff. I had covered maybe ten percent of what a textbook needed. <laughs> I was in this Donald Knuth conundrum that basically said, "Do I want to spend the next 15 years writing one, two, three, four, five, six, seven of these things?" The biggest piece of it that I had was basically the cost step model part. I said, "Why don't I just reduce the ambition level and do a software economics book?"

Basically, it introduced the COCOMO model. Since I was also teaching them things like, what should you be doing about calculating return on investments, and doing business cases, and things like that, those kind of things went into the model as well. Again, that was an outgrowth of what happened at TRW, but it became a more general kind of thing once it collided with the rest of the world.

Brock: How would you characterize the type of projects that the TRW software efforts, of which they were a part? I know there's a big satellite activity, there's a continuing missile activity. There are other broadenings of TRW's activity. Could you kind of paint a picture of that?

Boehm: Sure, yes. The defense and space part of TRW does have a San Bernardino office that covers what the missile people do. The Redondo Beach thing is right next to Aerospace Corporation and the Space and Missile Systems Center. San Diego does aircraft avionics, Huntsville did ballistic missile defense, and Sunnyvale did satellite control kind of things. And there's a big office in Denver that does similar kind of space kind of things.

Again, some of these were really opportunities to test whether the things that we were doing in Redondo Beach were really working as well elsewhere. One other interesting example, when we first calibrated COCOMO to the 20 projects, one project was an anomaly in that it was twice as productive as COCOMO said it was supposed to be. It turned out to be an offsite office in Dayton, Ohio, and our best conclusion was that if you live in Dayton, Ohio, the best thing that you can do on the weekend is program.

<laughter>

Boehm: And it wasn't really that way. It was they hired a whole bunch of really smart Purdue people. It was true that on the weekend they would get together for barbecues and talk shop. <laughs> Again, there were different situations in the different parts of TRW that you found. Another big win was that the ballistic missile defense people had some interest in getting better software tools. They gave TRW a whole bunch of money to build a software requirements analyzer and a design that mated with the requirements analyzer. That became RDD100 [remote digital display] and the ViaTech company that does those kind of tools.

It was fascinating seeing those different parts of TRW defense and space, but the other thing was that as the director of software research and technology, TRW was also a conglomerate that did TRW credit data. If credit data had problems, they would say, "Can you send us somebody over who can tell us something about configuration management?" Basically, I would be able to go and understand what was going on in more commercial kind of things. They had a similar group that was doing the communications processing and local area network kind of things.

They had gotten themselves into a problem in that they built a system that worked very well for one client, and then another client wanted something that was somewhat different. They would try to warp the original software into something that satisfied the next client and found that it was easy to be optimistic about how much effort it would take and that they would end up doing—they had a cowboy mentality. They had four people who could pull four consecutive all-nighters and deliver the next version for the next—but it was an untenable type of situation.

They had other people who were doing point-of-sale systems, and they were really good at engineering the point-of-sale systems and very good at marketing them, but they weren't very good at manufacturing them and integrating them and the like. Eventually we ended up partnering with Fujitsu who basically built the hardware for point of sale systems much more reliable than even TRW was doing. Again this got me involved in a bunch of projects that TRW was collaborating with Fujitsu on.

One project that I hadn't mentioned was that because of the Japanese rise in auto manufacturing, the other big part of TRW is the auto parts company in Cleveland. The T stands for Thompson for Thompson Products, and then basically he bankrolled Ramo and Wooldridge to become TRW. We would occasionally do things for them. For example, they started putting software into their carburetors and said, “We don't want any 5 million dollar or 5 million car recalls because we got the carburetor wrong. Could you send us some of your formal methods people who can mathematically prove that the carburetor software has no defects in it?”

Another thing that we ended up doing was building up a—because of this yellow peril the company said, "We've got to get better at productivity." Then something came down from Cleveland that said, "Every division has got to come up with something that will improve their productivity." My general manager basically said, "What can we do?" We did a study of, what were people doing to improve productivity, and what kind of strategies could we do? We found that you really need needed a mixed strategy. That there were a bunch of things that were really more personnel-oriented. If you could give everybody who stayed on the project for the whole project a bonus, if it's successfully completed, then we had less turnover on those projects than we had anywhere else.

We visited IBM and then found that they were investing in single-person offices and saying that if you keep your people in bullpens with the phone ringing all the time, they're losing flow and you should give them an opportunity to think and be more productive. We came up with a proposed software support environment for at least the missile and space division that I was in at the time, and did a COCOMO analysis that said, "If we can get our tools up from an average of nominal to an average of between high and very high, if we can get the personnel turnover to decrease and things like that, we should be able to double productivity."

Brock: Double?

Boehm: Uh-huh. We said, “If we believe the COCOMO model, let's give it a try.” The division general manager did get some extra funds to get the thing started and then said, "Well, you can pay for the rest of it with your productivity increase." <laughs> Again, we did another round of exploring difference support environments and workstations and the like. We came up to Xerox PARC and took a look at that, and said, "Ooh, that's wonderful."

Basically, they wanted $50,000 a seat, and the company said, "We can do $10,000 a seat, but that's—" there were a bunch of adequate Unix-based kind of workstations that we could put the rest of our software on top of. The other thing that our division general manager said we needed to do was to get him to identify a pilot project. It turned out that we had a 50-person pilot project that was just coming on board. Basically what we needed to do was to make sure that everything that we were doing was going to satisfy and help them improve their productivity.

So this was a really good thing, because I had hired a couple of people from UCLA and UC Santa Barbara that were very good at Unix, which was what we were going to use for the system. They turned out to be really good. But let's see, where was I going on this? Well, yes, they had things that they thought were really good computer science, but may not have been the things that were directly related to productivity. They would come up with automated test generation capabilities or things like that. We would have periodic meetings with the pilot project, and they said, "What we really need is something that when we change a document, puts a marker in the margin that says, 'Here's where the change is.'"

<laughter>

Boehm: This sort of deflated the scholars some, <laughs> but basically it did improve productivity and things like that. It was good have that project there to sort of keep us going. So we built it, and it doubled productivity.

Brock: It did double it? Wow.

Boehm: Yes. Subsequent <inaudible> projects did as well, in our division. It turned out that, unfortunately there were other divisions that were not as software-intensive but, yes, we want to do fancy graphics and we're going to have a fancy graphic interface for our support environment. After about five years, the super boss said, "Yes, we've got to merge these." Again, it turned out to be a really complex but satisfying thing, and the private offices worked fine. A good example was one of the programmers came out of his private office and said, "Anyone for lunch?" The neighbor said, "It's 4:00 p.m. You've been out there programming all day."

<laughter>

Brock: Well, it sounds like that wide-ranging across TRW while it rendered itself into a conglomerate was also a wealth of experiences for when you went to USC to teach. Such a wide exposure to all the different kind of problems people could face.

Boehm: Mm-hmm, mm-hmm.

Brock: But I did want to, before, I guess literally before you got to USC, you had your time at DARPA. I'd be interested to hear just how that invitation happened and what you…

Boehm: Well, let me just finish the USC thing.

Brock: Sure. Yes.

Boehm: Another thing I was fortunate to be a part of was Simon Ramo's Electronic Technology Advisory Committee.

Brock: Oh, wow.

Boehm: He basically included software as part of the electronics. So we would have monthly meetings that said, "What are we doing well, what do we need to do better, and what technologies have you been hearing about that the rest of us ought to know about?" At this one meeting, basically several people said, "We can't get enough software people." It's the end of the '70s and the end of the '80s and real estate prices are going sky high and the people in Iowa and Wisconsin don't want to come out to California and pay those prices. So, what can we do to entice more people to come?

One thing that Si Ramo was envious of—he had come from Hughes, and Hughes was owned by a foundation which had a lot of resources, and they used some of those resources to come up with a Hughes Masters Fellowship program, where people could get a master's degree at UCLA or USC in the kind of things that Hughes did, which were avionics and space and the like. TRW didn't have one of those, and so Simon Ramo asked me, "Can you go and work with USC and UCLA and see if they would like to have a special master's program in software and computing technology?"

I wasn't sure whether the universities would really be interested in yet another special program, that might want courses that they didn't teach and the like. But, in both cases they succeeded. I think for two main reasons. One is that all the students would be US citizens. Even then, we were getting more people from India and the like. The other thing was that Simon Ramo agreed to give each computer science department chair an annual donation if they would sponsor this program. We did it, and it was successful.

One of the things, though, that Simon Ramo said to me is that, "We don't want the people that come out of this to be just plain super programmers. What I found at TRW is that our best engineers are T-shaped people. They're deep in one discipline, but they also understand a lot of stuff about other disciplines. And the people that we would hire from bachelor's in computer science were very I-shaped." When I was reaching early retirement, I said, "Yes, what do I really want to do? Do I want to be here for another 10 years and retire at 65, or do I want to try to do something else?" What I decided I really wanted to do was to try to make I-shaped bachelor's in computer science people into T-shaped software engineers, and more system engineers, and that's what we've been doing.

We broadened the curriculum and put in human factors and user interfaces and management and economics and hardware engineering and the like, and then had this project course where everybody had to not just code something up that somebody gave them the requirement, but they had to interact with the clients and find out, "What are your needs and opportunities and what can we do in two semesters to build something useful for you?" That was about what I thought I was going to be doing in 1989, is early retiring from TRW and going to either UCLA or USC.

At that time, there was a DARPA program on advanced environments, and since we had built this productivity system, we were able to collaborate with actually, Lee Osterweil and Lori Clarke at UMass, and Dick [Richard] Taylor and Debra Richardson and the people at [UC] Irvine on, what is the next generation advanced software environment going to look like? What does its architecture look like? What kind of architecture validation tools does it need and what kind of user interface should it have? How is it going to address things like the Internet, and RESTful [representational state transfer] services and all sorts of exotic things that were coming along?

Basically, I became part of this program for a DARPA called Arcadia that Dick Taylor and Lee Osterweil, they were basically the two leaders of it, but TRW was bringing its piece of things to the party. So this got me involved with DARPA a lot more than I had before. We had done some things, like we wanted to be one of the first nodes on the Internet, and so we got TRW to put up the money to be a node on the Internet, the ARPANET.

The sponsors of these the Arcadia projects were two DARPA programmers: Steve [Stephen] Squires, who eventually became the chief engineer at Hewlett-Packard and Bill Scherlis, who went back to Carnegie Mellon and is now running their software engineering department. They were trying to find somebody to—the old director, Jacob Schwartz, was phasing out by the end of the summer and they were looking for somebody, and said, "Hey, would you be interested in doing this?" I said, "I don't have any grad students. I don't have any TRW projects. If I ever wanted to do something like this, this would be an opportunity, and it's got such tremendous leverage."

Not just the ARPANETs and software, but it had supercomputing, it had the world's most impressive network-based training facility for tank drivers that the world has ever seen called SIMNET [simulator networking], and a big budget basically. I spent a billion dollars of the government's money in the three years I was there, and spent the time basically being briefed by people who were the smartest people in AI, were the smartest people in robotics, or whatever, and my brain was fried by noon.

<laughter>

Boehm: But somehow or other, I learned a lot from all of these people. We were able to do some fairly significant things in the software environments area with building on Arcadia and the like. The other thing that Steve Squires and Bill Scherlis had done the year before was to get DARPA to give money to the Software Engineering Institute to set up a computer emergency response team.

END OF THE INTERVIEW