Oral History of David Cutler

Interviewed by:
Grant Saviers

Recorded: February 25, 2016
Medina, WA

CHM Reference number: X7733.2016

Grant Saviers: I'm Grant Saviers, Trustee at the Computer History Museum, here with Dave Cutler in his living room in Bellevue, on a beautiful Seattle day. Nice weather, for a change. I'm here to discuss with Dave his long career in the computer industry. And so we'll begin. So we were talking about your entry into computers. But let's roll back a bit into early childhood and living in Michigan and being near the Oldsmobile factory in Lansing. So tell us a little bit about your early days and high school.

David Cutler: Well, I grew up in Dewitt, Michigan, which is about ten miles north of Lansing, where Oldsmobile was at. I went to Dewitt High School. I was primarily an athlete, not a student, although I had pretty good grades at Dewitt. I was a 16-letter athlete. I played football, basketball, baseball and I ran track. And I'm a Michigan State fan. Michigan State was right there in East Lansing, so I grew up close to East Lansing and Michigan State. Early on, I was very interested in model airplanes, and built a lot of model airplanes and wanted to be a pilot when I grew up. But somehow that just went by the way. And later on, I discovered I had motion sickness problems, anyway, so that wasn't going to make it. So that's pretty much my early...

Saviers: So model airplanes kindled a technical interest in aviation and flying?

Cutler: Well, I don't know how I got into model airplanes. There was a local shop that was like a hobby shop that sold model airplanes, and I got interested in building them, and then, of course, you know, you build bigger ones and bigger ones. And you try to build powered ones, and I built a powered one, and I mostly crashed them. So I don't know if it really fostered a technical interest, but certainly was something that I was interested in for quite a while, but then sports took over in high school. And I became more interested in sports.

Saviers: So sports led to an opportunity to go to Olivet College. And athletics were a big part of making that possible, weren't they?

Cutler: I went to Olivet College, they offered me pretty much a full ride. I had an academic scholarship, and I had scholarships for basketball, baseball and football. And I would probably not have went to college, if I hadn't had a full ride, because we couldn't afford it. And I had applied a couple other places. I really wanted to go to Central Michigan. And they didn't accept me. And the thing at Olivet worked out really well. It was close to home. And that's where I ended up.

Saviers: And how did you decide on the course program that you'd take at Olivet?

Cutler: When I got there at Olivet, I didn't really know, I think, like most students, what I really wanted to study. I don't know why I decided on Math, other than it was a very interesting, challenging thing. And then after I decided Math was my major, then it was natural for me to say, "Well, I ought to have a lot of Physics," so I took a lot of Physics. And I really enjoyed Math and Physics. I didn't enjoy some of the other things. Olivet is a liberal arts school, so there was a lot of liberal arts requirements. It's also a religiously affiliated school, although really not tightly religiously affiliated, but there was a lot of stuff I had to take that was religious related. But I sort of gravitated towards the Math and the Physics, and it worked out well. It was great.

Saviers: So you're progressing through college and thinking about what's next, where the career might lead, where you might go to work. Any jobs during college, by the way?

Cutler: Did I have any jobs during college? The answer is No. I was so busy playing sports, and later on, the first two years of college I played mostly sports. And studied just enough to get by, and keep my academic scholarship. And then the last two years, I really, you might say, applied myself, and tried to be a good student. What happened to me was within a span of 18 months I had my right knee operated on, and they pulled out the medial meniscus. I broke my left leg, and then I had my left leg operated on, and they pulled out the medial meniscus there. So I was pretty much done with sports at the end of two years. And during those two years, I think that I mainly thought about-- I never thought that it was really going to end, that I was going to be-- I was thinking, "Well, I'm just going to be playing sports forever." And then after two years, it's sort of , "Well, I'm not going to be playing sports forever. What do I really want to do? And so I'm really not sure what I really wanted to do, but I think I really liked Math and I liked Physics, and I'm a hands-on kind of person that likes to build things. So I really think that I would like to have some kind of engineering job somewhere." So I was sort of aiming at an engineering job, but not going to an engineering school in a time era where it was possible that such a thing could happen. Today, that would probably be pretty remote that somebody could do that. I mean, if you had a Math degree, you probably wouldn't step into a mechanical engineering job or something like that. Of course, you could step into a computer engineering job. Computer engineers -- a lot of different disciplines make good computer engineers. So anyway, my goal in the last few years was to finish my degree and find an engineering job.

Saviers: And how did that go? Did you start interviewing companies? Did companies come to the campus?

Cutler: In those days, the companies didn't come to the campuses. We're talking 1965, and or at least they didn't come to our campus. And I just mainly applied to a lot of different companies. I applied to Lockheed, and the aerospace companies, and I applied to General Motors and I applied to DuPont. I don't know how I happened to apply to DuPont. I saw an ad somewhere probably, or something, and waited for people to come back and say, "Hey, we want to interview you." It turned out that I ended up actually having three solid job offers. One was from Chrysler in a soft trim plant, which was door panels, and the soft trim in an automobile is basically the inside. The job there was a time and studies operations research kind of job, in a town that wasn't too far from where I lived. I had a firm offer from Oldsmobile to work in their computer accounting department. And I had an offer from DuPont to work there in a job that was supposedly an engineering job. So I decided I didn't want to work Oldsmobile, because I didn't want an accounting job. I didn't know anything about computer science at that time. Computers were around, and people used them for record keeping, and accounting kinds of things. And of course, there were people at universities that were using them for other things. But mostly, when you talked about computers, the general public knowledge about computers was accounting and records keeping. And that's mostly what the IBM machines were doing was record keeping and accounting. So I decided I didn't want to work there either. And so I decided to go to Wilmington, Delaware, and take the job with DuPont.

Saviers: And was that job what you expected?

Cutler: Was the job what I expected? No. It was a life lesson in interviewing. And that when you interview and you decide you like to go to work for a company, you really should pin them down about what you're going to be doing when you get there, because if you don't, you may not be doing that. And I thought I was going into an engineering job. A little bit undefined, but I ended up when I got there with a job that was a tech writer, writing standard test methods for DuPont Elastomeric Materials Division, which were basically methods on how you stretch and pull and bounce synthetic rubber.

Saviers: That's pretty far from computers. What was the first step into computers?

Cutler: My first step into computing was with a group at the same lab I was at, which was a Sales Service Lab, which helped DuPont customers apply DuPont products to their products. DuPont basically made raw materials for other people to use in developing a product. For instance, the Elastomeric Materials Division made Neoprene. And Neoprene went into a lot of things like wetsuits, and even went into tires for a long time. So there was another group there which was a Math and Stat Group, and I'd interviewed with them, and they were interested in me. And after about a year, they asked if I could be lent to that group for a short amount of time to develop a model of a foam process for Scott Paper Company.

Saviers: Okay, so we were talking about a foam-making process at DuPont and some challenges in that that led to a computing career.

Cutler: So DuPont was trying to help Scott Paper company with a new process that they had developed to make foam for the garment industry. And the guy at the lab, the DuPont lab that was the manager of the Math and Stat Group wanted to build a model to help them schedule the production on this new machine. And so they asked if I could do that model. And I said yes, I'd do that, because I was so bored to tears trying to write these test methods, that nobody seemed to like what I was writing. So they sent me off to school with IBM to learn a language called GPSS-3. GPSS-3 was General System Simulator 3.

Saviers: General Purpose System Simulator.

Cutler: Yes, General Purpose System Simulator 3, which their third version of this. GPSS is a language that's an event-driven block-oriented language where you have a number of different blocks, or you could look at them as functions, that perform a specific kind of thing. Like Create Transaction was a block. So you could create transactions into the system that you're trying to simulate.

Saviers: And these all operated in parallel, too, right? It was a parallel event language, if I remember right.

Cutler: Yes, it is. The language tried to simulate a real-time process, a real time system. And then everything was running in parallel, but there was no need for synchronization. So you were removed from the synchronization aspect of building this model. And I ended up building a fairly large model, and part-way through the building of the model, I got really interested in more about how the computer worked than how the model worked. So I began to really dig into a lot about the computer architecture in the

machine I was running on. The machine that we ran this model was an IBM 4044-- 70-- yeah, it's a 44-- 4044.

Saviers: 7440, maybe?

Cutler: It was the precursor to the 7094.

So it was the 7044. That's what it was.

Saviers: Yes, 7044.

Cutler: It was a 36-bit machine. And this is the time that I really learned how fallible humans are. And we learned a lot of humility about programming, and the fact that computers only do exactly what you say, and not what you mean. And I went to the IBM Data Center. We had to go up to Philadelphia to go to the IBM Data Center, and with my deck of 2,000 cards with all kinds of time, free time,scheduled with the IBM representative and my boss and everything. And we're jollily going up there and we're going to run this model. And we get there, and we spent all the time trying to take the syntax errors out of the program, and we never got them out the first day. So that was a really humiliating experience.

Saviers: So the GPSS project finally worked, and you delivered the model.

Cutler: The GPSS project ran for, I think, about a year. And we had somebody from Scott Paper assigned to work with me that actually was somebody similar to me, very little experience and a young guy. And he and I would go up to the Data Center and run this model. And we would feed information into it and try to get results out, and what we were trying to do was, we were trying to figure out how, or what the best schedule was for scheduling a number of different colors and textures of foam on this machine to get the best productivity out of the machine. And eventually we got the model to work, and we said, "Okay, give us a question to ask the model," and they came up with a question about scheduling and we ran that. And we had a big presentation, and we handed over the model to them, and said, "Okay, here's the model. All you have to do is go to the IBM Data Center and run this, and you have a guy that can run this." And I really don't know what they did after that. I suspect they did very little with it, but it was sort of a-- a lot of it was public relations on DuPont having a vested interest in seeing that Scott Paper Company had the best support they could possibly get to run their foam process.

Saviers: So this initial foray into computers then leads to digging deeper into how computers really work, and how operating systems really work.

Cutler: So after this project, I got more interested in computing, and decided that I would really like to stay in the Math and Stat Group. So I stayed in the Math and Stat Group and worked on various things. We had an 1108. Well, it was 1107 to start with, a Univac machine at the Experimental Station, which was, I don't know, 15 to 20 miles away, and there was a shuttle that ran back and forth every day, and we could submit programs to run on that computer and get the results back. So I started doing some of that stuff for the Math and Stat Group, and also got involved with procuring a CDC-1700, which was a 16-bit machine, to automate the Instron machines, which was a tensile strength testing machine, for the lab. And so I got pretty interested in that. This all lasted, I would say maybe about two years, and then I

started to think about, that I really wanted to be somewhere where I was working more on computers, rather than working on computers as a tool to solve other problems. So I succeeded in negotiating a transfer to the Engineering Department at DuPont, which had lots of computing capability, and they're the ones that had the 1107, which they upgraded to the 1108. And I joined the Online Systems Group there.

Saviers: So the 1108 and 1107 had a product called Exec 2, which was notoriously flaky. <laughs>

Cutler: Well, what did I do in the Online Systems Group? Basically, I was trying to help the Engineering Service Division, which the Online System Group was in, was trying to help the Operating Departments, the Operating Plants within the Elastomer's Department-- actually with the Engineering Department, it's the whole company. So there was 13 Operating Departments. So they're trying to help them with their computer process control needs within the company. But there was a lot of downtime, because we were project oriented, and we're trying to go out and sell our services to somebody. So there was a lot of time when there wasn't much to do. So I got interested in the 1107 and the 1108, and I went to the System Programming Group and asked them if there was anything I could do for them. And they said, well, yes, I could probably help them figure out why Exec 2, which was the operating system to the 1108 was crashing so often. So they said, "What we want you to do is figure this out. And every time the 1108 crashes, we take a dump. And a dump is about, you know, four inches of a fanfold paper, which is, I think an octal dump. And we want you to figure out why, or what's wrong with the system." So at times I would have a stack of these dumps that was six-feet high in my office. And looking, at them trying to figure out what the pattern was for this. And I found several problems that were our fault. Our fault being DuPont's, because we made, I would say, about 2,000 lines of modifications to the Exec 2. But I never really did really figure out what was really wrong. And I guess an interesting side story is that later on when I was working at Microsoft in the early '80-- or I guess the early '90s, somebody from the University of Washington in Saint Louis, did a study on the timing of the 1108, and came to the conclusion that it would go metastable every four hours because of the synchronizers, and the system didn't synchronize all the clocks properly. So I guess that was the real answer to why it was so flaky.

Saviers: So this is kind of immersion by fire into what an operating system is.

Cutler: It is -- let me say, this was trial by fire. I was at the end of trying to figure out why something didn't work, which made me figure out how it worked. Right? So I had to figure out how it worked, to find out why it didn't work. And I'd be in these dumps, and I'd trace back in the dump, you know, maybe 1,000 instructions, and then the trail would just poof! It's gone. It's just gone. So eventually they got to the point where they trusted me enough that I actually became one of the main people that maintained the system, and made improvements to it-- what we thought were improvements. And put in DuPont specific changes to streamline the operation of the 1108.

Saviers: So this has been, "I now know something about operating systems, and I want to take my career further in that direction"?

Cutler: Operating systems really interested me. And I don't know why I decided that, boy, that was the way I wanted to go, but it was really the way I wanted to go. And maybe I didn't know it exactly at that point, but I sure-- some of the most satisfying time that I had was on the 1108. And then I-- the online

system part of it actually kicked in at one point, or a couple of points, where we did get some big things to do. We got a big job to work on at the Experimental Station, which was DuPont's main Research Center, which is a fairly big site. It was 120 acres and all 13 Operating Departments had offices there. So they decided that they would like to automate the analysis of some of their instrumentation. Their instrument data. And we proposed a Dual-PDP-10 system to do that. We spent, I don't know, maybe six months with various other vendors. IBM being a vendor, SDS, if you remember SDS? Scientific Data Systems being a vendor that we looked at, the Sigma 5. And we decided the PDP-10 was the best alternative. So we built, or we bought a PDP-10, a Dual-PDP-10 system with shared memory, special built by Computer Special Systems at DuPont-- or not DuPont-- Digital. And then we wrote an operating system for one of them, which was really the frontend. And it brought in all the data, and then when the data was complete it submitted it through the shared memory directly to the Tops-10 system which then did the analysis, of programs written in Fortran. And then we shipped the results back. So I got quite a bit of experience on DEC PDP-10s. Became pretty knowledgeable about the TOPS-10, and went to Digital to a school on TOPS-10, which was run by Dave Plumber, which was Digital's number one guy about running courses on the TOPS-10. So after that, I came back to the Engineering Department, and they gave me another job that was on DEC equipment, which was a PDP-11 gauge control system for Chronar film, which was used in magnetic tape. And this system was a PDP-11/20, and at that time there was no suitable operating system from Digital-- or not Digital-- yeah, Digital. No suitable operating system from Digital that would actually do this. We were going to do this on a very small PDP-11. I think it probably only had 8K of memory. So I was given the task of developing the little runtime, the little real-time runtime system to do the Chronar gauge control system. So when I finished that, I decided that maybe it was time for me to move on and think about something else. In fact, what I was thinking about was I'd moved from using computers to solve problems to actually working on computers themselves, but with not within a computer company. And I thought, well, if I really want to maximize my value to a company, and I want to be in the operating system business, then I ought to go to a company that builds computers. So I arranged an interview with Digital in Maynard, Massachusetts, and lo and behold, they hired me! So off to Digital I went.

Saviers: Just to backtrack a little bit here, what did you think of DuPont?

Cutler: I was at DuPont for seven years. And the culture at DuPont was really very foreign to me, because it was very-- it was an old company. DuPont was founded, I think, in the mid-1800s. They supplied black powder during the Civil War. That's how they got their start was black powder. A very bureaucratic company. Very regimented. You didn't tend to do things like go to talk to your boss' boss. So it was, I had a good time at DuPont. And it was great for my career. But I didn't feel as comfortable there as I actually ended up feeling at DEC, which was a very different kind of company.

Saviers: So did you have a sense of what the DEC culture was like when you decided that it was time to move to there?

Cutler: When I went to DEC, I had absolutely no idea what the culture was, other than the fact that I went to the PDP-10 programming course, so I was a little bit acclimated to the facilities. Digital was in a mill, and the mill was old. It led back to the mid-1800s also. But not really anything about the culture. And I really never really thought about that. I think I thought more about the opportunity than I did about, "Was

the culture going to be the same or different?" You know, if the culture had been the same as it was at DuPont, I still would have went, because the opportunity was better.

Saviers: So talk a little bit about getting started at DEC. Who was your first boss, and what was the direction of the task at hand?

Cutler: ,Let's back up a little bit about what I was hired for at Digital. This was like maybe another instance of not knowing exactly what you'd end up doing. Although, I really had this one tied down and I was going to work on something called OS-45. And OS-45 was a dream of Dave Stone, who was the head of programming at Digital. And the PDP 11/45 had some special solid-state memory that you could buy that was 300 nanosecond memory. And this is in 1971. And the PDP-10 was at that time, I think the best PDP-10 was above microsecond machine. So this was a very fast machine. And Dave Stone had this vision that we were going to recreate TOPS-10 on the 11/45, and we were also going to incorporate real-time capabilities, and batch capabilities, in addition to the timesharing. And there was a group of people that was assembled that was probably about eight people, and the leader of the group and my initial boss was Peter van Roekens. Peter came from RCA and he was responsible for the development of VMOS-- VMOS? Which was the Virtual Memory Operating System that ran on the RCA IBM-compatible machines. And so I came to Digital, and I spent probably the first two or three months thinking about all this stuff. And the group was thinking about all this stuff. And then all of a sudden, we weren't really making a lot of progress on this, because there was one little small problem that Dave Stone hadn't really considered as much as he really should have, and that was that the address space of the PDP-11 was 65K bytes. And 65K bytes just wasn't going to cut it for an address space. And the memory management capabilities weren't that great either. So we had an awful hard time really getting started and getting up to speed on what we were going to really do with this OS-45. So there was another project that was going on over in The Industrial Products Group. Develop a process control system, or a process control OS for the PDP-11. It was called RSX-11C. And they had been working on that for a couple of years, and it was going nowhere, and they asked me if I would go work on that. And I said, "Well, sure, I guess I will go work on that. You know, we're not doing much on this other project," so the first project worked on really as far as real work was the RSX-11C system.

Saviers: And the RSX-11 series ended up C then D then M, or something like that, if my recollection is correct.

Cutler: RSX-11C was the first in the series of the RSX systems, which were primarily sponsored by the Industrial Products Group, but were also sponsored by the Laboratory Data Products Group also, because they were in the real-time businesses, and RSX-11C was just a core only system that supported a few peripherals, A to D convertors, terminal interfaces. That was about it. And so there was a real need for some more functional systems. And as we filled out the PDP-11 line, of course the 11/45 was a pretty big machine, but we filled it out with a lot of other ones, the 11/40 and the 11/70. There was a need for some more operating system capability that supported a more rich programming and capability. And so there were several RSX systems. There was RSX-11C. There was RSX-11B, which maybe that one may have never had anybody really know much about it. But it was basically RSX-11C that could load DOS 11 files off of the disc. So it was -- we can start programs on the disc, and run them, but it's still a core-only system. And then RSX-11M, and RSX-11D. The order in which we did these was RSX-11C was done

first, and then B. And when I did B, I said, "I will do B, but I want you to promise me that I don't have to work on this again." And they said, "Okay." So about that same time RSX-11D was starting up, which was an equivalent of RSX-15, which was the 18-bit real-time system that Digital sold. And so I was assigned to work on that. And there was a fellow named Hank Kraichi, which was the main designer of that, because he worked on the RSX-15. And I agreed to do the overlay linker for it. So my involvement in RSX-11D was the overlay linker. And when we finished that that came out a little bit larger than we'd hoped, and a little bit slower than we'd hoped. So I had this idea of about building a system with a really different architecture. The RSX-11D had an architecture were basically everything ran as a process. And doing that meant the drivers ran as processes, which meant that there was an IPC, an Inter-Process Communication, mechanism had to exist between the program that wanted the service of a driver, and the driver. Which wasn't really conducive to the real-time performance we wanted to have. So I had this other idea of building a system that was a lot different. In fact, it's mostly like the other systems that we built, or I've been involved with. So I proposed building RSX-11M. And the Industrial Products Group got pretty excited about it. And they gave me, I guess, about eight people to work on that. And we completed that in about a year-and-a-half or two years. And that became the main real-time system for PDP-11s, because it would run on all the PDP-11s. The ones that had memory management system, or memory management units, and the ones that didn't. In fact it ran on systems from 32K bytes up to 2 megabytes, a big span.

Saviers: So this was the first time that you're running a major software project? A major operating system development project?

Cutler: Was this the first time that I was running a major project? Probably you would consider that to be the first time of a major project, because this was something that we're actually going to sell. We're going to sell RSX-11M. We're going to have customers, we're going to have support issues, and so there's all of that. And we're going to have documentation to do. The PDP-10 system was probably not quite as big a project, but I was the lead guy there, too. Very inexperienced. I don't know how DuPont ever had the faith they had in me to do that, but they did, and I'm thankful they did. So RSX-11M was really the first time that I did a project. And so I developed around RSX-11 a lot of things about, how are we going to do this, and what are we going to do, and what do we have to do. And so one of the first things I wanted to know is, "What is the scope of work? What are we going to do?" You can't just say we're going to do an operating system. We gotta know, "Well, what capabilities is it going to have? What are its constraints?" One of the constraints it had was it had to be subset compatible with RSX-11D. So it wasn't a blank sheet of paper where we could just go off and do a new system. It was, well, it has to be subset compatible, but it's source level compatible, not binary compatible. So that was a big caveat, because binary compatibility is much harder to achieve. So RSX-11M was really my first real operating system from scratch. I think it took about two years. It was delivered with what I think was pretty good quality for the tools and everything we had at that time. My philosophy about quality was quality was number one. It's still my first thing is quality is number one.

Saviers: So the "Size is the Goal" rubber stamp played into that?

Cutler: One of the big constraints with RSX-11M was, or one of the big problems with RSX-11D was it required so much memory. So with RSX-11M we wanted to build a much smaller system, and I figured

out right off the bat that the only way to do that was we give everybody a budget. So everybody had a budget about how much memory they could use. And so that was their target. So if somebody came back and said, "I can't possibly do that, and whatever the budget was, then we'd rethink that. But generally everybody met their budget. And so to reinforce that, every-- I had this rubber stamp made that said, "Size is the Goal." And everything was stamped with "Size is the Goal." You know, in those days, there was no email. So there was no electronic exchange of communication, so all communication was through memos, and every memo was stamped with "Size is the Goal." "Size is the Goal." So we ended up with in a 32K byte system, there was 16K bytes for the operating system, and 16K bytes for whatever the user environment was. Or the application environment. There wasn't really user mode. There was user mode and kernel mode, but these were dedicated systems. It wasn't a system where there were a lot of people who were going to use this in a general way. It was like, "This system is dedicated to controlling something in a real-time way." And whoever bought the system would write the software that would control their process.

Saviers: Now this is where the first Dave Cutler kernel, the DCK shows up? You go off and code a big part of the operating system yourself?

Cutler: Well, all the stuff that I worked on, I did a lot of it myself. I did-- going back to the PDP-10, I did a lot of that myself. In fact, everything that I've done throughout my career, I've always been a very hands-on person. I can't just stand back and say, "You know, this is what we're going to build, you guys go off and write the specs, and I'm going to manage you." I sort of become part of the team, and I look at myself as more of a leader than a manager. And I want to lead the people to build a product, and that means I'm a part of that. And there's no job that I can't-- that I won't do. I like to have-- I have this little saying that, "The successful people in the world are the people that do the things that the unsuccessful ones won't." So I've always been this person that, you know, I will build the system, I will fix the bugs, I will fix other people's bugs, I will fix build breaks. It's all part of getting the job done.

Saviers: So the PDP-11s run out of gas. DEC decides it's time to build a 32-bit computer. They're behind, the SDS and SEL and others have 32-bits, and a new operating system is needed, and the project VMS starts.

Cutler: About, I think it was 1975, Digital realized that they needed to build a 32-bit computer. And so being a part of RSX-11M and the 11 world, I kind of got involved in that. We had several architecture groups, several levels of architecture group. We decided to call this VAX, and VAX was a code name which stood for Virtual Address Extension, because the PDP-11 had such a small virtual address space. Gordon Bell was the person that was behind this. It was his vision that we should build this computer. The VAX-A was the hardware architecture group, and VAX-B was sort of a review group. And then I think there was another one called VAX-C, which was like the rest of the company. And I was like a formal member of VAX-B, but an informal member of VAX-A also. So I spent a lot of time on the VAX-A group discussing the computer and whatever. Bill Strecker was involved in this and Gordon Bell. This was my first real involvement with Gordon. He was at the company all the time I was there, but this is the first time I was really involved with him. And we would meet almost every day and talk about this computer and the instruction set and what we should do. And one of the biggest goals of VAX was to be able to encode a program in the same amount of memory that the PDP-11 could encode it. So that dictated an instruction

set that was pretty dense. The other thing is we definitely wanted to increase the virtual address space by quite a bit. And building a 32-bit computer would allow us to do that, and at that time, like going from 16 to 32 was like, "Man, how could we ever run out of 32 bits of address space." And later on, we'll skip forward and we'll figure out why we did run out of 32 bits. So the VAX-A and the architecture group, basically were defining the architecture. And there were other software people involved. There was some people form the PDP-10 group that were involved. Mainly Peter Conklin, and we hired a fellow from Xerox who had worked at SDS, who had worked on a system called CP-5, which was one of the systems that ran on the Sigma 5. It was the operating system of the Sigma 5. His name was Dick Hustvedt. Dick was involved with VAX-A, and the other person was involved with Peter Lipman. And some other people. But we basically developed the architecture and then the company said, "Well, let's build it." And so we said, "Well, how long do you want this to be?" And they said, "Well, two years seems reasonable." And he said, "Oh, my lord, we can never build this in two years, this is too complicated of an architecture to build in two years." So we had this off-site, one week, come to Jesus meeting about what we would do to the architecture to make this system buildable in two years, and that-- I think that was called the Blue Ribbon committee, and there were three software people-- three software implementers, and three hardware implementers, plus some of the architects. And we hammered out the final VAX architecture, which reduced the complexity of the instruction set architecture quite a bit. I suspect that people would still say that the instruction set architecture was overly complicated, but it was really complicated before that. And the other thing that happened was the memory management was so complicated that the only person that could actually explain it was Richie Lary. And everybody else was sort of like, "Yeah, that seems like it might work, but we're not really sure." And so we really simplified the management architecture so that the hardware architects thought that they could build it efficiently and the software people thought that we could build an operating system for it. And one of the decisions we made with VAX-- in fact we made a couple of decisions with VAX that limited the life of VAX. One was the encoding of the instruction stream to be equivalent to the PDP-11, which caused an extremely difficult to implement hardware system that was mainly in hardware and not mainly in microcode. So the VAX instruction set mostly was executed in microcode and not in hardware, because it was so hard to decode. And the other thing that was kind of a mistake, a long-term mistake was the page size being 512 bytes versus something more reasonable like 1K bytes or 4K bytes would have been a much better solution. However, we looked at it and the marketing people wanted to sell VAX systems that were the size of PDP-11 systems in memory. So they wanted to sell a system that was 128K bytes. Well, the most complicated instruction in the VAX architecture was "add pack six." That was an add packed instruction, decimal instruction with six operands. That could touch 40 pages. So it became, if it had to touch 40 pages, and you only have 128K bytes, then you don't want to get into a live-lock situation where what's going on in the system is such that you can never get all the pages in to actually execute instruction. So we settled on the 512 byte instruction set. So after the architecture was signed off on, and we decided there was a lot of fill there, we went through the blue-ribbon committee. Published the final architectural spec, then we put together the software group, and we put together the hardware group to build the hardware, and to build the software, and I was the technical lead of the software group. And then we spent-, I would say six months trying to think about and understand how to build the system, because we were deathly afraid of building a paging system. There wasn't a lot of experience on paging systems in those days. The PDP-10 did not have paging, it had swapping. BB&N had built a modification to the PDP-10, which had paging. But the scuttlebutt was it didn't run real well. Multix was a paging system, and Multix did not run well. I'm not sure

that Multix would run well on today's hardware. It was a great system, it spawned a lot of creative ideas, but it was basically very slow. Anybody that ran on that system definitely was not concerned about productivity. So we spent a long time figuring out how we would build this system, so we thought we would have good performance, we would be able to build it in a couple of years. And we would have the quality that we wanted to have. And so after six months, we authored a document, called it, I think, The Scarlet Design and Project Plan. Or it was the Scarlet Project Plan. And from that we started out and we implemented-- or actually we developed the VMS mainly on RSX-11M. RSX-11M had a little bit, by that time, it had a little bit of timesharing capability, so there was multiuse and we had a large 11/70. Probably two megabytes of memory on it. Two megabytes! So we developed mainly on RSX-11M but then we ran it on a simulator. The group that actually built the VAX-11/750 was the group that was assigned to build the emulator for VAX. And we actually had an emulator run on, which is a hardware emulator. We didn't run on a simulator, we're on an emulator. And we developed it-- got it running there. And then when the hardware, or the breadboard, for the VAX 11/780 was available, we took what we had and we went upstairs and it pretty much ran. It booted up and at least we got to the point where it didn't crash. So that's kind of how we started, and then from then on they built a prototype. I think it was Proto 4, and that was our development machine, and everybody ran on the development machine. And we ran what we built. And this was the first instance of the dog food thing. Where you run on what you're developing, and that gives you a big incentive to fix whatever's wrong. -- VMS is a timesharing system. So we were all in our offices, and all close to the machine, and if it crashed, you know, Dick Hustvedt and Peter Lipman and I would go out and look to see why it crashed. And we'd figure out why it crashed. And we'd reboot, go back to our office and fix the problem. And then go out and reboot a new system, and then everybody would be with the new system. So that's how the development of VMS occurred. And the day we shipped VMS, we had zero known bugs. That was zero *known* bugs. That didn't mean there was no bugs. But there was zero known bugs, and I know there was a bug right at the very end, where we had a race condition that was fixed like in the 11th hour. And then the final binaries were submitted to the manufacturing plant.

Saviers: So the VMS and the VAX 11/780 kind of launched DEC into the supermini, bigger applications, more nudging into the commercial world, all kinds of new features and requirements coming, and that spawns a lot of evolution of VMS. So you were with the VMS team for all of that.

Cutler: No. VMS was my first very difficult project where there were some constraints placed on us that were very hard. The two years that we were given to eventually or effectively produce a system of the level and caliber of VMS was very short. And at that point, I thought, you know, "Maybe I should try some other things I'm interested in." And one of the things I was very interested in was compilers. So at the end of VMS Version 1, I decided that I would start a compiler group to produce a PL/1 compiler. So I really didn't continue on with the continuing versions of VMS. I split off at that point and started this group consisting of, I think, four people. And I think we grew to about six people to do a PL/1 compiler, and we bought a frontend for the compiler from a company in Boston. And we produced the backend, which was the code generator. And I think that was another two-year project or so. And part of that project also spawned a C-compiler for VAX also.

Saviers: And this was an example of really eating the dog food, of writing the compiler in the compiler in PL/1?

Cutler: Actually PL/1, as far as the dog food kind of thing, was an interesting way we did the compiler. We had the frontend of the compiler that was delivered to us on Multix. And Multix, we first thought that what we would do is we would take the Multix compiler and we would compile the pieces of the Multix compiler and we'd bring them over to VAX and run them, eventually. We tried running it remotely on the Multix system, and we couldn't get it to run in full loop locks mode. And this is like, I don't know, a hundred, or whatever it is, it's fifty miles to Boston. It's 300 baud. So we tried half-duplex. It was a little better half-duplex. So we just, so that wouldn't work either. So we decided what we would do is we'd hire a shuttle service. And the guy who was working on the frontend would produce stuff on Multix, put it on mag tape, give it to the shuttle service, bring it to DEC, and we'd take it off the mag tape. So the first thing we did was we did-- we bootstrapped the compiler backwards. So we were doing a code generator on our end, but the code generator was mostly written in assembler. So we got that working, and then we would bring back the phases of the compiler one by one, backwards until we got back to the beginning of the compiler, and then we could compile the whole thing at Digital. So we really were eating our dog food, because the whole compiler was written in PL/1 except for the parts of the code generator that had to run last.

Saviers: So now we're about 1981 at DEC or so? Or a little earlier?

Cutler: This is about, well, it's 1980/'81, sometime during that year. Maybe early in that year.

Saviers: So some circumstances conspire and you think, "Maynard's no longer for me."

Cutler: So what happened after the PL/1 compiler was, I getting really a little bit fed up with the growth of Digital. When I started at Digital, there were 12,000 people, and I don't know in '81, there certainly wasn't 12,000 anymore. There was 75,000 or something. And so what started out to be this really engineering run, engineering environment, suddenly became more of a management, marketing, and program management environment, which I wasn't too happy with. And so I had expressed some things to Gordon about maybe that I would like to be somewhere else. And he had offered to-- he said, "Well, anywhere you want to go, you know, maybe we can figure out how we can start an engineering facility there." And I said, "Well, I don't know. Maybe I'll be okay here." And then this opportunity came up to start a company that was going to commercialize a compiler-- I'm sorry, a language, that was funded by the government for a real-time control, which was called Praxis, and it was done for Lawrence Livermore. And it was actually done for the Shiva Nova project, which was a fusion energy project. So I kind of got enthused about doing that. And we-- "we" being, I guess, about six people. Three of us from DEC and a couple from-- or four of us from DEC, and a couple from Lawrence Livermore. You know, traveled around the Bay area looking for money, and we got to the point where I think we figured out that maybe this wasn't the right thing to do, or at least I figured out this wasn't the right thing to do. And so I told Digital <earlier>, "I'm leaving." And so, you know, gees, we had a termination date and everything. And then all of a sudden I think, "Oh, you know, I'm not sure this is going to work out, you know?" The President of the company was going to be a guy from the Shiva Nova project, and he seemed to be more interested in how he was going to spend all of our seed money, which we had put up ourselves, on things like his

colors, and paying his wife for our stationery. And so suddenly all our money that we'd put together was gone. And I'm thinking, "Wow, I don't know if I want to go with this guy." So at the last moment, I said, "Nah, I'm not going do it." So effectively all the other DEC guys says, "Well, we're not doing it either." So then became the question about well, "Gees, what are you going to do?" And so I said, "Well, you know, I think it's really appropriate that I think about being somewhere else where I'm kind of isolated a little bit from-- a little smaller environment, where we don't have all of the bureaucracy we're developing at DEC." So Gordon says, "Well, gees, that sounds good to me. Where would you like to go?" And I said, "Well, I can't be on the East Coast, because that won't be far enough. You know, I really don't really like the Southwest too much. So, maybe like Arizona and New Mexico. Probably Texas is not a good place. Probably the West Coast would be the best place." Well, Gordon says, you know, "The West Coast would be fine, but I want you to be somewhere where there's a university that's got a good Computer Science program, so that we can draw people from there, and maybe we can actually collaborate with the university." So the DEC real estate department arranged this trip where we would fly to several different places on the West Coast and look at. And we went to San Francisco, we went to Sacramento. They were really hot on Santa Rosa. We went to Portland, and we came to Seattle. And I had spent a little time in Portland, a little earlier. In fact, what had spawned the earlier offer by Gordon to establish an engineering facility for me was that I had actually interviewed at Intel in Portland, which is kind of an interest-- this is kind of a backtrack, but only go back and talk about that just for a second. And this happened and I interviewed and they were building the 432 at that time, which was their 32-bit computer. Which eventually never ever materialized. But we had lost a couple people. Dave Best, who had been the program manager for the 11/750 went there. So I interviewed out there and it got to the point of they were going to offer me a job, and they said, "Well, we can't really offer you a job, because we really cooperate a lot with Digital, and you've got to tell Gordon Bell that you're interested in this." So I said, "Eh, no problem." I called Gordon, I said, "Gordon, I've been interviewing with Intel. And I don't really want their job, but I want their offer, because I want to figure out how much I'm worth." I said, "Okay. You okay with that?" And he says, "Sure." And I hung up and about 15 minutes later, the phone rings, and it's Larry Portner's admin. Said, "Larry and Gordon would like to come up to Spit Brook Road in New Hampshire and have lunch with you. And I said, "Oh, that's fine. When?" "Today. Will that be okay?" And I said, "Sure." So they came to Spit Brook and we went to lunch, and they basically said, "Well, why do you want to leave?" And I said, "Well, I don't want to leave, I just wanted to know what my value is in the market." They said, "Oh, well, maybe we can work something out." I said, "I really like Portland. And Gordon said, "Well, maybe we can think about this." So, you know, they did something about the salary thing, and something about stock options and I thought about that a little bit. Then I said, "Hey, I don't really want to do that. I'll stay here for a while," and then this other company opportunity came up. And then when that fell through, it was really like, "I really do want to move." So anyway, we're back to the West Coast, we're going back to the West Coast, and we go everywhere, and we come to Seattle, and it's raining, but everything's green. And so I said, "Gee, boy this is really a nice place." And we visited the university. We talked to the Chairman of the Computer Science Department. I forget what his name was. He was really interested in us coming. And so we went back home and we thought about it for a while, and said, "Seattle's gotta be the place. That's got to be the place." So that's how we ended up in Seattle. Sort of we took the trip and looked at some different places, and decided we didn't really like Sacramento very well. The Bay area was a little bit too busy for us. And Bellevue was a pretty nice place to be. So that's where we ended up.

Saviers: So here you are. So a number of people decided to come with you, and some other folks were hired here locally. You started up a lab.

Cutler: We came here and we brought, I guess about eight people with us. And we hired-- you know, one of the things was we liked to hire some people from the University of Washington to firm up that connection with the University of Washington. We hired a couple people from there, and started on a project - our charter, by the way, for this was to produce -- again, going back to the real time roots-- to produce some software for embedded chip-based VAX systems. We were working on a chip set at that time called Scorpio. Which by today's standards would have been huge, because it was six chips. I think three of them were custom, custom VSLI chips. So it was a big compared to today, but it was very small compared to what we had in those days. So our charter was to build this runtime system, a little real time operating system, which was very easy to program. And which was very easy to implement real-time applications. And the name of the system was VAXELN. And it was my first entry, our first entry into multithreading. We actually did multithreading in VAXELN, which is about the same time that the Mach kernal was doing multithreading at Carnegie Mellon, just about the same time. So we got partway through this project, and we started thinking about, "Well, Scorpio has been going for quite a few years now. We don't know when it's going to be delivered. We don't really have any suitable platform to run this on. Maybe we should think about what we should do to VAX to maybe make VAX a smaller architecture." So we started an effort within the company called MicroVAX, which was to trim down the architecture. And then later on, after we had developed the architecture, we thought, "Well, we still don't have any platform to run this on, because we're going to have to produce a chip for this." And there was a MicroVAX chip. "So maybe we should start a hardware group here, too, and we should produce an implementation of MicroVAX that would not be one chip, but it would be a small system, maybe the size of a PDP-11/05. So we brought some more people out from DEC-East, for the hardware project. And they were mainly people from the East, but we also hired, I guess, three people locally, because there's a lot going on in this area with electronics, so it was easy to hire a few people, and to produce MicroVAX-1. We actually shipped VAXELN and MicroVAX-1 together. And I got very interested in hardware at that time, and I ended up working on VAXELN pretty much writing the kernel. And the other people wrote the pieces of the file system, and the pieces that went around it. But then I switched to the hardware group for the MicroVAX-1 and wrote all the microcode.

Saviers: And somewhere along the way, you hooked up with Carver Mead and took the team to learn how to do chip design.

Cutler: The way that I implemented the MicroVAX-1 system was Gordon thought it would be a good idea if we tried some new technology with respect to producing silicon chips. And one of the new technologies that was being touted by Carver Mead from, was he from Caltech?

Saviers: Caltech, yes.

Cutler: And Lyn Conway had some silicon power tools and the company that was founded-- actually called Silicon Compilers that Carver Mead was a principle in, and John Hennessey was also. And we got together with them, and inspect a chip that they could build that would be the central processing unit of the system, and it would be on one chip. And so that's-- they produced the chip. They designed the chip

jointly with our specifications going back and forth. They didn't have all the expertise. And then our hardware guys designed all the hardware that went around that. We had an awful time getting the chip built. We went to a company called AMI to start with, which, you know, in those days semi-conductor companies were popping up like dandelions, there were lots of them. They'd pop up, they'd run through their VC funding. They'd roll over and play dead. AMI never actually produced a chip for us that worked. They produced a chip and we got the chip and we looked at it, and they had done all the masks backwards. They were all the negative of what they should be.

Saviers: Oops.

Cutler: Oops. So I don't know how that happened. So we went to another company called SEEQ. And SEEQ actually produced the chip. I can't remember us having any problems with the chip with respect to errata. There may have been some things that we worked-- I don't really remember. I remember it being pretty much bug free. And we were able to take that chip and we put it in a Q-Bus-based system. And that was MicroVAX-1, and we ran VAXELN on it, but later on, we also ran VMS on it. Part-way through the MicroVAX architecture discussion, we had trimmed the memory management capabilities of VMS quite a bit. Not VMS, the memory management capabilities of VAX way down, which meant that VMS would not run with it. It would have to be big modification to VMS to make it run. And so I was writing the microcode and I thought, "Gees, why don't we just do the full memory management, there's not much microcode." So I got Bob Supnik to agree to that. He was the guy who was doing the microcode for the MicroVAX-1 chip, that we'd just do the whole thing. It was going to be less work to do that, than not do it, and we get VMS to run. So we got VMS to run almost at the same time we got our system to run. And I don't remember exactly the year, it was probably '86 or something, maybe '84. About three years, maybe three years in development for that system. So that was our first project at DEC West.

Saviers: And where did that lead to?

Cutler: Well that led-- what did that lead to? That led to problems! <laughs> That led to big problems, because here we are, now we have this kind of seasoned operating system group. We have the seasoned hardware group, now what are we going to do? And so the problem is to try to find something to do that's not being done by another group at headquarters. And so we went through several proposals on building projects, or building systems hardware, was mainly hardware. There were some people at DEC West that wanted to build a workstation. And then there was some people that wanted to build a more capable system than the workstation. And we ended up not building the workstation. And we ended up trying to build a bigger system, and that led to big problems, because we were always over the top. We were always duplicating some function that-- or some part of the price performance space that somebody at headquarters was already doing. So we went through several projects trying to find the right VAX to build. And, basically, I would go back to Maynard and try to negotiate what we should build and get some degree of buy-in and come back, and we'd work for six months and then Maynard would decide, well that wasn't what they wanted us to build. So we'd have to find something else to build. So this sort of led to us thinking about RISC was becoming very popular at that time, and it was kind of getting to the point where there was a lot of publicity about RISC having better performance than VAX and RISC was the way to go, and so we got interested in RISC. And there were several RISC projects going on in the company. There was one going on in Marlboro. There was one going on down in Bay area at WRL,

which was Western Research Lab. And we were sort of working on one. So Jack Smith, who was the Senior Vice President of Engineering said, "DEC West is responsible for the RISC architecture." So there started the Prism project, which was a new RISC architecture for DEC. And I think we probably started that in about '85, I think maybe? '85? So that's what happened after we were-- we were very successful at being that, we're just tremendously successful. Very low overhead. Very few people. We got things done in just record speed. For record amount of money. Engineering, NRE costs were like zilcho compared to the other projects. But then into a period of not being able to find the right thing to do, and then into the architecture. The new architecture.

Saviers: So I guess it was RISC war that started to evolve at DEC between various parts of the company?

Cutler: Well, the reason that the Prism architecture actually got assigned to DEC West was because there were so many projects going on in the company trying to develop RISC architectures. And of course, Digital already had-- we now had the 36-bit line, which was the PDP-10, and then the DEC System 20. We had the VAX line, we had the PDP-11 line. I don't think we were probably making any 18 bit 9s and 15s. We may have still been shipping a few 8s by then. But so we really didn't need three or four new risk architectures. So the Prism architecture started out much like the VAX architecture with representatives from-- VMS had a representative. Hudson, which was the semiconductor branch had a representative. Marlboro had a representative. One of the guys that worked on that architecture was there. We had myself, and then we had a guy from the architecture group. And we then spent the next three years, just about three years, going through a revision of this, and then taking it back to the senior technical community. And they would look at it and say, "Oh, we don't like this." And we would go back. We flipped from 64 to 32 to 64 back to 32 in the architecture, which it took a long time to sort through all that, and then the whole principle of RISC was to produce a system architecture that was easy to implement. You didn't have to have brute force methods of doing things. And massive verification suites. But then the company kind of ran off the track and started adding all these things. The first thing that happened was we had to have vector arithmetic. So we have to compete with Seymour Cray. So we had to have a vector architecture. Well, Seymour probably had, you know, he probably had eight vector instructions. Well, we had to have 150 vector instructions. And they had to be long vectors, they weren't short vectors. So this went on and on and on. And you know, in the meantime at DEC West, we're sort of gearing up-- you think we'd be in a holding pattern, but we're not, we're sort of gearing up for building one of these systems. And we have proposals for building these systems, and we took MicroVAX-1 and we actually converted it to be-- to run Prism code, so people could actually develop software. And we came to a point where finally the company just said, "We're cancelling Prism." And I went through several-- we went through a lot of iterations. I'd go to Maynard, try to get some consensus on what we should be doing. I'd come back home, and invariably it was always something bigger and we'd add people. So we kind of grew from like 20 people to 180 people. So we had this 180-people organization, and we're building kind of a nebulous product based on this new architecture at a time when the company was trying to build three of these machines. There was in Marlboro and one in Littleton and one in Seattle. So our machine got canceled. And our operating system, which was called Mica, got canceled. And I sort of got-- I kind of went through the stages of trying to find some ground where we could all agree on what we were going to build, come back. I'd sell the people on it, and then they'd get disappointed. So I said, "Eh, I

don't want to do this again." So I said, "What am I going to do?" And I said, "Well, I guess I'll form a company." So I got three other guys, and I, together, these are the same three guys we met with when I was talking earlier about Ballmer, which we'll get into later. And we decided we were going to form a company to build servers. And I formally quit, although I didn't leave right away. They gave me an office somewhere else, until I had really settled things. I spent a lot of time travelling back and forth to the Valley talking to the VCs and what we're doing and whatever. And we actually got to the point where we had the VCs all lined up. We had three VCs and MIPS. We were all going to put up some seed money. We were going to have a million dollars' worth of seed money, which was probably quite a bit of money in today's dollars. And this would have been in 1988. So somewhere during this period, a quite a few of the people, when Prism was canceled, had decided that maybe they'd look across the street at Microsoft, and go to Microsoft. And one of the people that went over there had been talking to Bill, Bill Gates, about the group at DEC West, and that we had a good operating system group, and maybe that would be something that he might be interested in. So there was an interview arranged and I went over and talked to Bill about what they were interested in doing, and what they wanted to do. And see if I was interested. And their basic concern was that Intel was not bringing the X-86 architecture forward fast enough, and that RISC processors were going to overtake them, and if RISC processors overtake them, then all the PC software that they developed wouldn't be valuable anymore. And that they ought to have something going on in RISC. And if they were going to have something going on in RISC then they needed a new operating system that wasn't written in assembler, that was written in portable language, so that they could run across several different architectures. And so I sort of listened to all of this, and was really interested in doing my own company. And so I kind of left the interview thinking, "Well, I don't think I'm real interested in this." And sort of, you know, time kind of-- or you know, a couple of months, we mulled this all over, the four of us. And finally we got all the VC lined up and we got the commitment from them for the money. And before we were-- we got the commitment-- boy, you listen to them talk, we were the greatest people in the world! But as soon as we got the commitment, our value seemed to diminish a little bit. And they were a little bit more vocal about whether or not we could actually run a company or not, and whether we'd be able to develop a product, even though we had a good history of product development. So that sort of got under my skin a little bit. And about the same time, Steve Ballmer, who worked at Microsoft, and he was the Head of the Finance Department at Microsoft, called and said, "You know, what are you thinking?" And I said, "Well, I'm thinking that we're really close to doing this company, and we're probably going to turn down the-- or I'm probably going to turn down the Microsoft offer." And he said, "Oh, gees, why don't you give me a chance to convince you otherwise?" And I said, "Well, yeah, okay." And he said, "Well, how about breakfast at Denny's, you know, on Saturday morning at seven o'clock." Or I don't know, maybe it was a Sunday morning. I mean, but it was early. And I said, "Yeah, okay." So the four of us went to Denny's, and sat down. And Steve is such an energetic guy, and a positive thinker! And so he starts the sales job, and I guess the best way-- thing you can say is he fleeced us! <laughter> Because by the time we left, we were all ready to sign up. And we did. We decided not to form our company, which I think was a great-- looking back on it, we would have been just another server company probably, unless we could have developed some-- in fact, what we were really founding was a hardware company. And unless we would have had the foresight to convert that company quickly to a software company, we would have been in trouble. So in fact, an interesting thing, you might find interesting here, is we actually were going to run UNIX on this box, because of the lead time that it would take to actually produce something of your own. And the fact that it would be incompatible with everything. So obviously, we couldn't produce

something that's VAX compatible, so the next best thing was UNIX. And we decided that Mach was the best example of a UNIX system. So we were actually going to use Mach on this server. But anyway, Ballmer fleeces us, and we all decide to sign up and go to work for Microsoft. And so, you know, I quit immediately. They're still working for DEC. And like I quit on Friday, and joined Microsoft on Monday, and the following Monday, these other three guys follow me. And then from there, we-- I don't want to say we recruited, because we didn't really recruit people, but people asked us about opportunities. And of course, we're forthcoming about opportunities, so we managed to bring over quite a few people. And part of the thing about going to Microsoft was they wanted the portable operating system, but I like the hardware part of it, too, having that. So we convinced them, or I convinced them that I should have a hardware group and a software group, and that we would produce-- the hardware group would actually produce the first RISC-based PC that the software group would be producing the operating system for. So we were able to bring over-- I'm sorry, we weren't able to bring over, we were able to satisfy the--

Saviers: I think the statute of limitations has long passed.

Cutler: The statute of limitation long-passed. We were able to bring over these people from DEC that had been the top performers at DEC West, and so we got the top hardware guys, and we got the top software guys. It was about 10/10 or something like that. And we hired some more people. We hired-- we had, part of the thing about the group was that we-- the hardware group, not like the software group-- the software group, Microsoft wanted to make sure we had Microsoft people in the software group because otherwise we would just produce something that was DEC, the DEC cultural thing, right? So we had, actually I think we had one, one software guy, and he was a pretty good-- or he was a vocal Microsoft guy, and he had a lot of ideas. Which, actually dovetailed with ours pretty well about how to build an operating system. Microsoft had been working with IBM on OS/2 for several years by that time. And that relationship was not going real well, and this guy had worked on that, so he was very happy to join our group.

Saviers: So you parachute into Microsoft, and after a short period of time you look around, what did you see? What did Microsoft look like at that time?

Cutler: Well, Microsoft didn't look like DEC. Or actually I should say, Microsoft was a very different company than DEC. DEC was-- the thing I liked about DEC, and I think that I can honestly say that I spent most of my best-- the most enjoyable engineering years at DEC between 1971 and 1980, or maybe '82 or '83-- was DEC was mainly an engineering company. It was run mainly by the engineers. The engineers proposed the products. They talked to customers. They tried to produce very high-quality products. And put them in the market. And then the marketing arm of DEC was not-- I would say during those years was a really strong. I think it became strong after that. Much stronger, because the focus of the company became much more marketing-oriented than it was engineering oriented. But Microsoft was very marketing oriented. Microsoft was-- while they marketed the PC software DOS to IBM, so they were very more marketing oriented, which, you know, I thought was-- I liked that, because-- I liked it and I didn't like it. I liked it because it was going to be-- I was going to have a group of marketing people that were really going to be out hawking the projects, or products. On the other hand, I didn't like it because they might have their fingers in what we were going to do. And it turned out that the original version of NT was pretty much an engineering-driven thing. Except for a couple of things and that was that in addition to

being portable, Microsoft wanted the system to be compatible with multiple operating environments. So that sort of led us to an architecture within NT that would allow different subsystem personalities. And so we had to keep IBM happy and run OS/2. Of course we want to run DOS. And then the other thing that was big was very popular. Big, was popular, was POSIX. These are the standardized UNIX interfaces. So the personality of this thing was going to be this multifaceted thing. So, let's see, I'm trying to think. Why did I go down this road?

Cutler: Oh, okay, we were dropped in.

Cutler: So anyway, the Microsoft people, or Microsoft sort of dedicated that level, but none of the rest of the levels. The rest of the level was pretty engineering-driven. And we set about designing Windows NT much in the same way that we had set about designing VMS. Except that in terms of Windows NT we decided to produce a much more detailed specification of the system. In VMS, we sort of had-- we had this small group, and we spent a lot of time talking about things. And we did do some specs. But they were pretty rudimentary specs. For NT, we wrote a humongous spec. And I would say it was over six months, about how the whole system would work, so that people could pick up the spec. As new people came in, they could pick up the spec and understand it and read the code and actually kind of understand what was going to go on. Our first projected delivery date for the OS was like two years. And that actually took us five years to produce the first OS. I could have never-- if you'd ask me even today if it would have taken five years back then, I would have said, "No," I'd say, "We could do it in two years." But there was so much more to do. VMS did not have a graphical user interface. We had to do that. We didn't have the subsystem problem in VMS. We had a running multiple personalities. We had to run the PDP-11 binaries. And we produced what was called the AME, which was the Application Migration Executive. And we ran all the-- in fact, that's how we got the VMS software out was by running the 11 software on the VAX.

Saviers: Completely unconstrained architecture in the I/O, and so on, in the PC.

Cutler: Right. And so we had a lot more to do. And we started out with this idea that we're going to produce the first RISC PC and we're going to produce the software for it, and we make this-- well, I can't say that we make this mistake. This mistake has already been made for us when we get there is Intel at the time is producing the i860. And this is one of the fastest RISC machines in certain applications. And those applications are mainly graphics. So Microsoft wanted to use that as the microprocessor for our first reference implementation. We were going to build a reference implementation of the 860 and we got one of the OEMs, which was Olivetti, that signed up to build this. So we were going to build the reference system, and they were going to take the design, make it manufacturable, and manufacture it. We were going to build the software. But then part way through that, we found out that the i860 was pretty sick. It had a lot of errata that would make it pretty unusable as an OS-based system. It had some quirks that were-- made it very nice for the graphics world, but not very well for the operating system world. Had a virtually tagged cache. So basically every time you switched context, you'd sweep the cache, because it has all the virtual tags from the last address space you ran in. So anyway, it took five years to do the first version. So I guess the original question was, "How different was the environment?" Well, the environment was a lot different, because DEC was an engineering run company, and were always based on engineering excellence. Microsoft, I think, justifiably, I would say, is a marketing-based company. And they weren't based on engineering excellence. So there was sort of a little clash between our culture, and

the Microsoft culture. But I think that we instilled a lot of that into the Microsoft culture, but we still had a lot of things going on that were sort of outside our control that were within the Microsoft culture. So it made the project a lot longer, and the environment-- it was very different.

Saviers: Somewhere along the way, you said to me, "There were a lot of coders at Microsoft, but no software engineering." And you had to bring the discipline of software engineering to Microsoft.

Cutler: Yes, I think one of the things that we brought was the engineering discipline on building software, and I don't think that Microsoft had the engineering discipline that we had at that time. Of course, you know, that's 28 years ago. That's a long time, and the discipline at Microsoft now is, I would say is excellent. There's a lot of stuff in place. It's not the same atmosphere that we had with the original NT team, because our team was small. You know, we started out with 25 guys. At the end of Version 1, were 150. At the end of Version 4, were 400, which is big-- today it's 4,000.

Saviers: Now along the way, a lot of things get added to what NT is. I guess you had Program Managers, that was, I think, the title.

Cutler: As far as the culture and environment, at DEC we had Program Managers, but Program Managers were sort of-- they were quasi marketing people in some sense. They didn't try to manage the engineering effort. The engineering effort was managed by the engineering people. So there was a big clash in the beginning between me and the Program Managers of Microsoft over who did what. And I wrote-- I wish I could find this piece of mail I wrote. But I wrote this big thing, big piece of mail to our Program Manager about his job and mine, which laid out what I thought his job was, and what my job was. And so that pretty much worked itself out, because we pretty much operated the way we did at DEC. But over time Program Managers at Microsoft had taken a more direct involvement and control over what happens.

Saviers: So you have this enormously complicated thing. Multiple platforms you're going to run on, portability. Keeping control of this, and keeping everybody moving for five years gets to be quite a challenge.

Cutler: So a little bit about the complexity of what we're building. We talked about the 860, and the problems with the 860, well, part-way down the road, we'd only went about a year, and we decided there's no way we can go with the 860. And so we said, "What should we do?" So we said, "Well, how about MIPS? MIPS is doing the R4000. That's the platform that we ought to make the first one." So our hardware groups switched to the R4000, and we started to go in that direction. Now, just to make a lot of difference to the operating system, because the way the operating system is structured, or was structured was we're sort of taking the stuff that was absolutely portable, and it was still portable, and then the stuff that was hardware architecture specific was divided out so that you just had to replace that. So the first thing that happened was DEC got wind of what we're doing and says, "Well, gees, we're building an R3000 workstation. Why don't you support the R3000, too?" So I guess, foolishly, we said, "Okay, we'll support the 3000, too." Well, that put us into some difficulty now, because the 3000 is not entirely compatible with the 4000, so now we're building for two platforms. So we had the R3000 and the R4000. And not long after, said, "Well, it looks like it's going be a little while longer than we thought. Maybe we

should add the 386 back into the mix here. You know, Intel's coming out with, I think it's a 386-25, 25 megahertz." <laughter> Let's add that in, so now we start-- we aren't building any hardware, because there is the industry infrastructure to build x86 base system. So we don't do anything there. And so now we're building for basically three targets and three operating environments. And the GUI stuff. And now along comes DEC back with Alpha and says, "Gees, we want to run Windows on Alpha, too." And so we said, "Okay, so we started doing Alpha." And then IBM says, "Well, gees, as long as you're doing all that, let's do the Power PC." So we're doing actually four platforms. We're doing four platforms if you only count MIPS as1, if you count as 2, then we're doing five. In addition, what happened was somewhere in the early '90s, we had this thing on the 16 bits- that ran on DOS that was a layer. It was a program that ran on top of DOS, it was called Windows. And we did Windows 1.0, the industry said, "Ho-hum." We did Windows 2.0, the industry said, "Ho-hum." We did Windows 3.0, the industry said, "Ho-hum." We did Windows 3.1, we sold 16 million copies in six months! It was like unbelievable! So Bill says, "Hm, I guess our main operating environment will not be OS/2 anymore, it will be Windows." So we switched from our main operating environment being OS/2 with the Program Manager, which was their graphical interface to Windows. And when we brought in Windows, then we brought in also all the compatibility requirements with Windows 16. So we're running on all these platforms. We're running these different environments, we're running a 16-bit DOS emulator. We're running a 16-bit Windows on Windows emulator. We're running Windows 32-bit emulators on Alpha, Power PC and MIPS. All those things to get to the first release. Now the first release actually only supported MIPS and the x86. And I think about six months later we did another project called Daytona, which was another release. And that included Alpha, which was 3.5, and then about six months later we did the Power PC. So that's kind of why it took five years.

Saviers: Easy to understand in retrospect. Lots of things changed.

Cutler: Lots of things changed.

Saviers: So keeping the team going, there's the Dave Cutler infamous management style at work. Talk about keeping the team going.

Cutler: I said earlier that I've always considered myself as a leader and not a manager. So I've always tried to be a leader that is part of the team, versus somebody that is separated from the team. So I'm always a part of the team. If the team is stressed and up against up the wall, then I'm stressed and up against the wall. So I think we didn't have any problems with people burned out. You know, our people pretty much stood the whole five years. Although, there was some ramifications. I mean, people got divorced, you know, there were other things that happened. And we had very little attrition. I mean, just really small attrition. Everybody believed in the product. And the other thing we did was we did this-- every Friday night we had a party. We called it the WIM. And the WIM was the Weekly Integration Meeting. And it was instigated by a guy named Dave Thompson, who was in the Network Group. Because we eventually pulled in the Network Group. Originally in the PC world, the Network Group was separate from the Operating System Group and it was just an add-on. You know, it was just-- it's some other program. If you want to do network operations, you ran a program to do that. It wasn't building the operating system. And we could plan all along to build it into the operating system with the VMS model. So we're growing pretty rapidly as we're pulling in some of these groups and we're going from 20 people to 150 or whatever. And so he said, "Well, why don't we have this-- we'll have a meeting every Friday

afternoon and we'll go get beer and pizza and chips, and we'll meet, and just everybody will get to know each other." So this grew into a huge thing! I mean, I can imagine five years and every Friday night that you're doing this. We start out, we're in a room about this size, and we end up in a huge room! And we start out that I send the development managers, there was like four development managers under me, I send them down to Safeway to buy a few beers and pick up some pizza and bring them back, to the end when we have full catering from Marriott. Marriott's bringing everything, and we go from like chips and salsa and beer or pizza, to shrimp and all sorts of stuff that's catered. Kegs of beer. I mean, it was interesting. And then there was some things that were, we'd always have something going on where we tried to get people to feel better about what's going on, and how much stress they're in. I know that there's some people that felt they were under a lot of stress, and sometimes they'd break, but I think generally we held everything together pretty good.

Saviers: So blowing off steam with the team. Skiing, racing.

Cutler: Well, one of the ways we blew off steam a little bit was in fact the guy that was the Microsoft guy that had joined us early on, decided that it would be a great idea to go down to Laguna Seca and go to the Russell Racing School. So I think about maybe eight of us went down to the school and we went to the beginner's school, which I think was four days. And we were in a Mazda open-wheel car. And you know, that was a good thing to blow off a little steam. And it was really an eye-opener about, for me, about driving a racecar. I really wasn't very good at it. Or wasn't as good as I thought I should have been at it. And the beginner's school, they just teach you basically how do drive the car, about the dynamics of the car, G-forces, cornering and all that sort of stuff. But then they have an advanced course where you actually at the end of the course you get to do a race. So a few of us decided, "Well, we'd like to go to that, too." So, I don't know, a couple/three months expire and we go down and we do that. And so we do the race. We go through the advanced school, and they do some more things. And I'm getting a little better at it. And we finish that, we do our race, and that was fun, so I thought, "Well, maybe, you know, they have the School Series, what they call the School Series." And you basically go down once a month, and you race on Saturday, and then on Sunday, and then you come home. And Debbie[1] was involved in this, too. Debbie came down and did the beginner one and she did the advanced one. And so she and I would go down there every month for, I think we went down from, oh, somewhere in March to September, something like that. Every month, so it was ten races or something like that. So I did that. And that was fun. And so then they had-- every year they had what they called a Pro Race. And the pro race they would-- they didn't have governors on the systems, but they sort of would tune-- they would open them up a little bit. They'd reprogram the ECU, the Engine Control Unit, and give them a little more horsepower. And they'd say, "We're going to Pro Race now," and the Pro Race meant there was a prize package, and you could win some money. I think probably the top was 10K or something like that. But this would draw a lot of drivers that were pretty good. So I did one of those. I think I finished next to last, only because one guy, my first one, only because the other guy was worse than I was. So then I got involved with a guy that was one of the instructors down there, and another guy that owned a race team to import an Atlantic car from New Zealand. And that was when I started into the racing really seriously. And so I owned that car, and he raced that car one year, which was 1995, and the next year I decided that, well, we really should

[1] [Editor's note] Debbie is Dave Cutler's wife.

have a new car to race. So I bought a new car, and so the question became what to do with the old car. So I climbed in old car, and gees, I was pretty good. So I decided, "Well, I'll race in this series." So I started racing in the Atlantic Series and I raced 56 races in that series. And had some pretty horrific accidents. Nothing that I was really hurt really badly in. But one at Laguna Seca is sort of an infamous one, because ESPN played this over and over and over again for, I don't know, five minutes of broadcast time or something. And one of the turns I did a longitudinal flip. So you think about a flip as going end-over-end. It was a barrel-roll flip. I did a flip like this, and landed on the top. And it was in a corner where it was really dusty and the dust flew, and there was big dust clouds.

Saviers: What turn was that?

Cutler: It was Turn 10 at Laguna Seca.

Saviers: So a fast downhill sweeper--

Cutler: To the right.

Saviers: To the right, yes.

Cutler: And ESPN picked that up, and they just played that-- they played it over and over, because there's a certain amount of time they have to-- the yellow goes out, they have to clear the track, then they have to get the wreckers out and upright the car. So the race is,

Saviers: Stopped.

Cutler: Stopped. But they just keep playing this over and over and over again. <laughter> So that was sort of my big wreck. My best result, I finished eighth one time at the Milwaukee Mile. This particular series, we raced Monterey, Mexico, which was a real-- that was a real eye-opener. The race tracks here are pretty safe. The racetrack there was not real safe. I raced in Homestead. I raced in Cleveland at the airport; Mid-Ohio, which is outside of Columbus; Montreal; Trois Riviera, which is up by Quebec City; Vancouver, British Columbia, Portland, Long Beach.

Saviers: Road America?

Cutler: oh, yeah, Road America. In fact Road America. This in Elkhart, Wisconsin. Yeah, I had a big wreck at-- my second biggest wreck, or maybe my first biggest wreck was at Road America. There's a turn there they call The Kink, which is about 140- or 150-mile per hour turn. And I lost control of the car and hit a stone wall there. And--

Saviers: Ouch.

Cutler: I got out of the car, and the car was rocking like this, and I couldn't really understand why that was the case, and I looked down and all four wheels are gone. <laughter> I'm just sitting there in the car, you know, the car is built with what they call a monocoque tub, and I'm in the tub, and the engine's still in the back, but none of the wheels, they're all gone. So I did race at Road America.

Saviers: Your favorite one?

Cutler: The Milwaukee Mile, by far. Nazareth, Pennsylvania. I raced there, too.

Saviers: Any analogies from racing into software development?

Cutler: Yes, actually, I think there is. It's focus. I've always been a person that really can focus on the job at hand. And racing is-- driving a race car is focus. I mean, what happened ten milliseconds ago, it's gone. You're going to focus on what's coming up and what's going to happen. So I think there's some analogy there about being really successful, I mean, in anything, it's to focus on what's important, and what you have to achieve to get to the success. So in racecar driving, it was just another example of that.

Saviers: So the super intensity of concentration.

When you're writing code, how do you achieve that? Can you stand music being on, or want to be in a quiet office?

Cutler: In general, noise doesn't bother me much. I would prefer that somebody's not-- there's one guy at work now that whistles. And that's not a problem, but he whistles down the hall, and I'm like, "I really don't like that." But generally noise doesn't bother me much. But I do focus on what's going on, especially if I write a new piece of code, you know, I'll write the code and I will spend maybe three or four iterations of executing the code in my mind. All the error paths, all the paths it can go through to see if I can prove that it's correct, before I even run the code. And most of the time when I run code, it pretty much runs the first time. It may not be absolutely correct, but it will run. It won't crash. So that's being able to focus on what's going on, and not have things bother you or whatever.

Saviers: In the NT program, it was demolishing bugs. How would you change the process to create less errors and make them easier to fix?

Cutler: The easiest-- probably the easiest bugs to fix. well, the question is how to make bugs easier to fix and--

Saviers: And fewer of them.

Cutler: I think fewer of them is probably the easier of the two, because the fewer the bugs are, the harder the ones that are left are going to be solve. And the hardest bugs to solve are the ones around synchronization. They're not around correctness of code flow. They're around the correctness of synchronization, where there's not the correct barriers, there's not the correct locks, or there's race conditions between code paths, or something. One of the things that also exacerbated the five years was multi-processing. I had the bee in my bonnet when I went to Microsoft that we were going to produce MP systems. And nobody else was producing MP systems. DEC had, at that time they had the VAX 782, which was this funny asymmetric thing. And we had an 11/70 that actually was SMP, but there was some Unibus problem that we decided not to manufacture that. So SMP problems, synchronization is a big problem. The problem with less bugs is, I think, through the engineering process itself, and the people-- you got to have-- quality has to start from the bottom up. It can't start from the top-down. You can't

legislate quality, quality has to be something that's inborn in everybody at the lowest level. And any link in the lowest level that doesn't produce a quality piece produces bugs up the line, because they interact with other pieces of the system. If you're building the lower levels of the system, then your leverage as far as causing problems up in the levels is very high. So I think reducing the number of bugs is basically thinking more about quality and paying more attention to quality. I don't think that necessarily you have this thing where, "Well, we'll do 100 percent test afterward. You know, we'll do everything and then we'll test to see how good it is." It's got to be built in to start with. So I think building systems with fewer bugs means paying attention to that all the way along. And not-- for me, I hate it when there's a bug filed against me. I don't like that at all. And I don't want that bug to exist filed against me for any longer than I have to. I'll drop everything to go fix that bug, because I know I can fix that. So there needs to be that kind of mentality. I mean, there's a lot of people that say, "Oh, yeah, I got 50 bugs. Oh, you know, I'll get them fixed." Well, it's like quitting smoking. You know when the best time to quit smoking is? Right now! It's not an hour from now. Or a day from now, or a week from now. It's right now! So the best time to fix the bug is right now. So that would be fewer bugs. Make them easier to fix, I don't know how we'd make them easier to fix, other than building better tools for analyzing what's going on, I guess, is primarily the way we would make them easier to fix. And we have lots of tools now. We have lots of tools that do the same thing. You know, that somebody has said, "Well, that's a good tool, but I'd like the tool to do this. I don't own the tool, but I think I could build a tool that was a little better that would do this, and maybe we could find something a little better." And so we have a lot of tools. We have a lot of source level tools. One of the big problems that we had partway through the evolution of NT was hitting a big security attack or wall or whatever. Somewhere after about 2001, somewhere in there. The internet, we really got going on the internet somewhere in the late '90s. And the internet suddenly had problems with people with viruses and trying to attack your system. So we developed some tools that analyzed the source and tried to find places where you have buffer overflows - are the biggest one. And the amount of buffer overflows that we had was just mind-boggling. And it was like we have this tool, and the tool says, "Look it, right here, this could overflow." And so that was great, I mean, that made it so that it lowered the bugs. Even things like, I mean, an arithmetic overflow can cause a bug because maybe you're calculating the size of the buffer that you want to allocate, because you got a byte count, and you do this computation and, gees, that overflows. And now it looks like you need a small size. So you allocate that size, and then you use this other size to transfer data into it, and all of a sudden you have a problem there, so you have some tools that analyze the source, and actually point out places where there's bugs. And we actually fix those like they're bugs. The problem with those tools are that they're static analysis and they have a lot of false positives. So you see a lot of things that aren't really problems. And they can't see some things. You may have, in your code, you may have had logic that made it impossible for the thing that it's complaining about to occur. And it'll say, "Hey, this is bad." So those have to be filtered out. And so we have ways to filter that out, too.

Saviers: If you go back a little bit to getting programmers to really produce good code, is there a Dave Cutler School of how to get a programmer moving along in that direction?

Cutler: Do I have a way to move people to build good code? I guess the only way that I can say that I have-- help people build good code is by example. Everything that I do, I try to make it my best. And everybody sees the source. And obviously they see that I don't have a very big bug count, or a zero bug

count. They look at my code, and I hope that that really helps them become better programmers. But another aspect of this is really with the young people starting out to really engrain in them the importance of quality. I mean, for me, there's nothing worse than software that doesn't work. I mean, I just can get really upset with software that doesn't work. Where it's obvious that something just-- I mean, it's something simple. I mean, if something doesn't work on your phone, or it doesn't work on your tablet, or it doesn't work on your PC that should work, it should infuriate you, because it just should not be that way, right? It should be-- all the stuff that doesn't work should be stuff that is little tiny things. Maybe it's like there's a few pixels over here that are wrong. Or something that's just minor. In fact, a lot of the bugs that are-- that have been and continue to be filed against NT are these kind of things. They're cosmetic things. And they tend to-- they have low priority. So those things are not upsetting, but it's upsetting when you have something where you ship a product and it's in the field for four months, and all of a sudden there's this disaster that happens where you wipe out somebody's data, and then all of a sudden, it's like, "Wow, that's really a problem." So I think quality is something that starting out with young people and saying, "Hey, you know, we really got to have quality."

Saviers: So we come to '93, I think the final release date of NT? Or the official release. Millions of copies shipped.

Cutler: The initial version of NT is shipped in 1993. And then we shipped, NT 3.5, which was Alpha support about six months later. And part of that was also a kind of cleanup, let's improve the performance of the system release. We did not have the performance level in the system in the first release we would have liked to have had. It was a little bit bloated. Now bloated in those days was like maybe it took 24 megabytes. <laughter> As opposed to two gigabytes, or four gigabytes. So Daytona was kind of this dual thing where it was this-- it was called Daytona. It was actually the Daytona release, and our big thing there was the racetrack and everything. And I think we actually may have-- well, we had some racetrack stuff, because we'd been all off to the race school. And we incorporated Alpha support. And then just a little bit later than that, we did the Power PC. So that brought us to the end of the first release, and then the next release was NT 4.0. And NT 4.0 was an interesting release, because at the end of the Release 1, or we actually called it 3.1, because Windows was 3.1. The 16-bit Windows was 3.1. And the feeling was that nobody would buy an operating system that was a 1.0. Nobody. So let's start out with a number that wasn't a 1.0, and, well, we'd pick the same number as the 16-bit thing. So then we had 3.5 and 3.5.1. And well, the next one was going to be 4.0. And in the meantime, another guy had been hired from Banyan. His name was Jim Allchin, and he had actually-- was starting another project called Cairo. And Cairo was envisioned to be a system that would incorporate some of the things you now see with internet search. Only it would be locally. A new file system that had indexing. Indexing all your mail and everything. And it was internally in Microsoft, it was called "Information at your fingertips." And he started this project, and it was kind of running in parallel with the tail end of Release 1. So when we started Release 2, well, actually it was NT 4.0, the Cairo project. Allchin actually became my boss. And the Cairo project became part of the NT development. And that was supposedly where we were going to be going. And I was really worried about this, so I proposed another project, called-- and we called it Tukwila. And the reason we picked Tukwila was because we know where Tukwila is and we know how to get there. <laughter> It's just down the street. So I convinced Jim that we should-- we had the capability to do both of these at once, and throw one away. And it turned out that the one we threw away was Cairo. And NT

4.0 was probably one of the best systems that we produced. Certainly not functionally. But as far as its acceptance and quality and performance. It contained the-- it was in 1996, so it was just after Windows 95. We'll talk a little about that lineage of system. So it contained the new graphical user interface, which was drastically different from 3.1. 3.1 was sort of, it was graphical, but you sort of put things together. You built your menus and you had to do a whole bunch of stuff. And then the UI actually had all that stuff. You know, everything, when you got the system was already installed with Start Menus and all that stuff. So NT 4.0 had the new UI in in, and it was an evolution of NT. Cairo would have been an evolution of NT also, but it would have had a new file system, a new shell, just a whole bunch of new stuff -a new mail system, everything. So I started to divert our-- I digress into Windows 95, and why Windows 95 came about. Windows 95 is actually built on the DOS codebase. So it was actually the 16-bit system with the capability to run 32-bit programs, and it included this new UI. And it actually was very popular and sold a lot of systems, so in that timeframe, it sold a lot more systems than NT sold. But NT sold in the places where you were doing enterprise stuff. You needed some security, you needed to run server-- you wanted to run file servers or print servers. Those are the only kind of servers we had then. So NT was alive and well, and very healthy. But at the same time Windows 95 made the company a lot of money. And then they continued on with that same codebase for Windows 98, and then with Millennium, which I believe was in 2000. So then it died. It completely died. It was a non-extensible-- it was bug-ridden. It was notorious for just hanging. It'd just freeze. And that was not unexpected, because it was built on DOS, it wasn't built on--

Saviers: And terrible security.

Cutler: No security.

Cutler: None. None. One of the things-- you didn't ask me about this, but I want to say something about security. You know, we started out in the beginning to have really robust security. And we designed the original security model was pretty robust. But by today's standards completely different. Because in those days, -- you were protecting people against file access and system resource access. But it wasn't like the security today where somebody invites the breech right into the system. They're in the browser, and they click on something, and here comes a program they didn't even know about. And all of a sudden it exploits a buffer overflow, and bang! It puts something in the operating system, and now you're under the control of the virus, right? So anyway, there was a lot of controversy about whether we should make people logon or not. Because the original PC, you know, you turn it on, there it is! Well, Bill didn't want a logon. We're talking about security and the fact that we maybe shortcut the system a little bit with security in the beginning because the PC mentality was you just turn on the system and use it. And we really had-- we had come from the background of timesharing systems, where everybody logged on and everything was shared. And you were worried people accessing your files. And gees, if it's a PC, you own it, you can access your files. So we actually designed the security system and built the security system, and we did have logon, but there were some aspects of the system that we didn't do quite as well as we could have. One was when you install an application, you should really install an application for the user that installed it. Not for the system. But since it's a PC, you install it for the system. So that was one problem, because now you assume now that everybody is going to use this thing, and that if two people have two different versions, well, you can't have that. You can only have one version. And then the other thing that

happened with that was we would have all these people that were reshipping binaries. And so somebody would have an application like Office, and they're reship a binary, and somebody would have maybe a database application that would ship a binary. Same binary. Never been tested against the other guys' product, right? So we had-- and we're still somewhat trying to work through this about state separation, which is part of casting this as part of security is separating everybody's state, so it's only their state. But NT does have security, and it had security built-in, whereas Windows 95 did not.

Saviers: So the end of the DOS Windows comes - it can't go any further. And where does NT go in terms of influencing the future of what's called personal operating systems?

Cutler: Where does it-- oh, you mean like the--

Saviers: XP, 7, and so on.

Cutler: Well, let me tell you, it's all NT. There is nothing that's not NT. Win 10 is NT. It's all the same stuff. It's an evolution of the same codebase going all the way back to 1988. There is some interesting diversion along the way. If you want to go into that right now, or it's sort of a little bit later, because I was going to start--

Saviers: Take the direction you want to take.

Cutler: I was going to say what happens after NT 4.0.

Saviers: Good.

Cutler: After NT 4.0, NT has grown to 400 people. And I'm sort of at the point of maybe I don't want to manage 400 people. Maybe I want to just work on the technical things that are important. So I decide at NT 4.0, that I'm not managing the system any more, that somebody else is going to manage the system. And I become an individual contributor. And being an individual contributor from then on, basically meant I was doing the same things I was doing on the coding implementation side as I was doing before, I just wasn't carrying all the other stuff with me. So the next system we produce is Win 2K. Windows 2000. And Windows 2000 was probably one of the first systems that we really produced-- as a separate server system, server SKU from a personal SKU. I don't know that we produced a server SKU, we probably produced a server SKU for NT 4.0, too, but the 2000 was more oriented towards that, about writing server applications. Win 2K was another great system as far as the reliability and everything. And my part of it was just basically any new features we put in the kernel, performance, nothing real big. And then we shipped that in 2000 and in 2000 there's a guy that's in charge of the Consumer Group, and there's a guy that's in charge of the Server Group. They've been working together under another guy to produce the system. But the guy that's in charge of the Consumer Group decides that consumers don't need the quality that the server group needs. And that the Server Group's laundry list of features that they want to put in the next release is way beyond what they need for the next consumer release. So we split the codebase. We split it in XP, and the other one was the Server 2003 codebase. And that turned out to be not a good decision. And what happened there was in the Server 2003 codebase we fixed all the security problems we were having. We put in the server features, and it actually turned out in the end, it almost

shipped at the same time as the XP codebase. And the XP codebase shipped with some new UI stuff. It had the new Start Button, and some other stuff. It was mostly UI work. But also shipped with a lot of security problems, and more bugs than the Server 2003. And I worked mainly on our Server 2003. And about this time-- I don't remember the exact day, but it was slightly after 2000, AMD came to us and said, " We have an extension to the x86 architecture that will extend it compatibly to 64 bits. Are you interested?" And one of the things I kind of left out here is the addition of yet another platform in-- which was in Win 2K, which was the Itanium. We had an Itanium support in Win 2K, and the Itanium would run x86 binaries, the 32-bit binaries, but it wouldn't run them at speed. So there was a speed penalty. Plus the fact that the Itanium was never, ever as fast as the fastest x86, ever. So AMD proposing this extension looked kind of good. And so we sort of looked at it and said, "Well, it looks pretty interesting. You know, why don't you go back and come back when you've got it more solid, and you think you would want to build it." So they come back in like six months and say, "Here it is." And I look at it and say, "Boy, this is great! The x86 runs at speed." We can produce a small, like the AME thing, we can produce a small wrapper envelope that just executes 86 binaries, and they'll run it whatever the speed of the processor is. The 64-bit stuff will run at the speed of the processor. Except it'll have a bigger address space." So I said, "This is pretty good. We need to talk about whether we're going to do this or not." So I don't know, not very much time went by, and there was not much happening. And I went to Allchin and said, "We're doing it. We're just going to do this. I'm going to do it. I've got a couple guys, and we'll do it." So four of us started out to do this. And we had done a lot of groundwork on making our codebase datatype neutral, as far as pointer size, because of the Itanium, and also because we were actually at one time doing a 64-bit Alpha system. And then Compaq said, "We're not building those anymore," so we quit doing that. So we started out on what became the x64 version of NT in the early 2000s. At the same time, AMD was developing the architecture. And we got a pretty good say in some things that were really important. I think we got about six features put into the processor architecture that I consider to be one of them was absolutely essential, and that was they put in relative addressing. And it's 32-bit displacement, so it meant that if we were willing to say our binaries were never any bigger than two gigabytes, then that displacement would always span a binary. So they put in the relative addressing. They put in some other things that were performance oriented. And I had a really good relationship with the lead microcode guy, I mean, Kevin McGrath was just great. Going back to how important this 32-bit displacement thing was. Maybe you wouldn't think that would be important, but one is position independent code. It's hard to write code for the 64-bit architecture that isn't position independent. Which means that one of the main security things that you do is something called ALSR, which is Address Space Relocation Randomization. So you take an image and you randomize where you put it in the address space. We have to relocate it when you do that. Well, if there's no relocations, you don't have to do much, right? Now there still could be relocations, but to have position independent code, you need the code to be independent, as opposed to the data. You can always produce pointers to the data. So anyway, that, we decided to do that with the Windows 2003 codebase. In the meantime, the XP has shipped, and we're producing a system on the consumer side called Longhorn. I called it, "Does it Matterhorn?" Because it became so bug-ridden and so hard to actually get it run, that eventually it got to the point that I convinced Allchin, I said, "We should just junk that codebase, and start over with the Windows 2003 codebase, as the new codebase." And we did that. We started over. That system became Vista. And Vista was-- the way that that got delivered with system-- was Vista. But we actually delivered the x64 support as a Service Pack to the Windows 2003 Server. It was Windows 2003 Server SP-1. And it had some new SKUs. And they were 64-bit SKUs.

There was a professional desktop SKU and then there was a server SKU. And that turned out to be very successful. I mean, everything's 64-bits now. Almost everything. Well, I can't say the telephone is not 64-bits, because--

Saviers: Cell phone is anyway.

Cutler: Well, the cell phone, it turns out is actually going to 64-bit ARM. So they want more than 4 gigabytes of physical address space. So they're going to go to 64-bit ARM. There is some places where I thought the 32-bits would entirely go away. But there are some things like cell phones, where smaller pointer sizes actually do end up meaning less memory. You use less memory. And on phones, less memory means less power drain. Less power drain means happier users. So in the meantime here, we ship 2003. But just before we ship 2003, we have to go back to XP again. And we have to spend a ginormous effort on XP to bring it up to all the security enhancements we've made to the Server system. So basically we do XP Service Pack 2, which was basically a release. A re-release of that software. So that sort of brings us to the point where-- and again, this is all NT. You can read, like you read Mary Jo Foley-- I don't know if you know who she is? She's written about Microsoft forever. And it's like, "This is a *new* operating system!" Well, it's not a new operating system. It's an evolution. It's like every time Linux comes out with a new release, it's not a new system, it's an evolution of the old system. So the XP SP-2 was really a new system, because it was a re-release of almost every single binary. And then we went back to did XP Service Pack 2, then we did the x64 release. And then we went back and put the resources back on Vista. And so Vista came out basic-- as a 64-bit system, and 32. I think Vista actually supported the Itanium, also.

In the meantime, one of the things we left out here was the RISC systems were dropping off. And here's why I think they dropped off. I think the RISC guys were so focused on simple architecture with simple implementation Intel got this architecture, it's a little complex, but they got lots of transistors, they've got lots of design, and they got lots of verification expertise, let's put these transistors to work to make everything run faster. And running faster is things like very sophisticated branch prediction, better caching. You know, set-associative caching. And just all sorts of stuff, speculation, and all that. But the RISC guys are like, "Well, you know, we don't really want to do that stuff. You know, we want this to be simple." So the thing about RISC having a performance advantage just kind of went away. It went away because there was nobody that had the resources to put to work on the chip architecture and the chip design that Intel did, and they didn't have the resources to do the big designs with all the transistors they had. I mean, they're building billion transistor chips now, right? So, and the RISC guys are going-- they want to be able to say, "See, look at this! We built this machine with 300,000 gates." Right? Like, "So what?" And the stuff on the Intel processors now and the AMD processors is really complicated. It really is. The out of order execution, and all this stuff. And the RISC guys, you know, they don't want to do that. So I think that's why the RISC systems kind of went away is that they couldn't offer the price performance that the Intel systems could. They couldn't provide the compatibility through the line. I mean, we had all these software emulators running. And one of the things that we're doing now in Itanium, and actually Alpha started this, too, is binary recompilation. Where you take a binary, you take an x86 binary, you run it through a binary recompilation, produce another binary, which runs under a special emulation system that mostly executes at speed of the processor. Because it's translated everything that it can find from

x86 code to Alpha code, or whatever the target is, but there still may be a few pieces around there, and so you actually run the real binary. And sometimes you have to actually flip into a mode that actually emulates the instruction, but most of the time it executes. So binary recompilation has come a long ways, and we would be a lot better off. But still you have--- the binary recompilation and then you say, ok-"I'll do the recompilation and it'll just put it in the store, right? It'll work then." Well, it doesn't, you know? You have to go and make sure. You have to test and make sure. So it's a doubling of the work. So I think the RISC architectures went away in their popularity because of all those things. Now ARM has taken a hold in phones, and it's taken a hold in some of the mobile devices. But ARM is not anywhere near the performance capability of the x86 stuff, either from AMD or Intel. And you know, ARM is trying to get there. They're trying to up their performance. But they've been primarily aimed at power. And you use ARM processors in applications where you don't need a lot of power. Maybe you need a little chip. One of the things that's happened is Intel and AMD to a certain extent, is like, "We got a billion transistors. We just got to build this chip that's this big. The last one was that size. The next one's gotta be that size. Even though we have four times as many transistors, we're going to build it the same size," right? Well, the ARM guys actually haven't done that. ARM guys have a portfolio of a lot of different sized chips. So if you're building a watch, you can build it with a chip that's smaller, right?

Saviers: Yes

Cutler: But Intel and AMD have never take this thing about, "Well, let's take-- we now can build the same functionality in one-quarter the size; maybe that would be a good product."

Saviers: Well, this whole semiconductor process technology with the end game with the P4 is we get to maybe 4 gigahertz, and the chip is incandescent. So now we go to multicores. So this is a big event in the operating system world that we're going to have four processors to play with. How did that start coming into the environment?

Cutler: NT was from Day 1 we built NT to be MP. And our initial, our RISC prototype, or our reference, was MP. So we were MP from Day 1. So cores are no different than-- there's two kinds of cores, actually. You think about cores as being autonomous cores, or multithreaded cores. If they're multithreaded cores, then they're a set of context that's a thread, and the core hardware that actually executes that context. And then you switch among these contexts. And Intel did, it's called SMTs-- well, multithreading. Just call it multithreading. Intel did this early on, but we already had MP support. So the only thing that was kind of a curve ball there was that SMP increased your parallelism, but not necessarily your performance. Because you basically have the same amount of execution capability that you had before, you just have multiple contexts that you can run against that. Now the place that you could gain is that there's delays, like cache misses, where you could say, "Well, that cache miss-- I missed in the cache on this set of context, I can switch to another context," and it can immediately execute. Maybe you had a cache miss before, and now that data's there. So you can get, maybe 15 percent. You can also lose five percent, or ten percent. So very unpredictable, so that kind of threw us a curve ball. AMD went the other way, and they did what was-- what they called CMT. Multithreading was called SMT, Symmetric Multithreading. And the other was called CMT, and CMT was Chip Multiprocessing. Which actually put more than one processor on a chip. So the original AMD 64-bit chips were two-processor chips. So they had two real processors. Now, since then they've all migrated to this multicore thing where they actually have core

logic that shares some other piece. They may share, like you may have, a couple cores that share an L2 cache, for instance. Then all the two pairs share on L3 cache. And all of that now is all in NT. How many levels of cache do you have? What are they, how many ways set associative are they? And all of this stuff. So the thing didn't really affect NT particularly, but it affected the applications. Because the applications now could get more performance if they're multithreaded. So if somebody started out originally with NT and said, "I'll just have a single-threaded app," then these other processors-- well, they progress through to the point where we're at 3.5 gigahertz, and they're all ready for the next leap to 4, and the next leap to 4 doesn't happen, because we can't build it at 4, because we would have such high current leakage that you're right, it would glow. The thing would glow. So we started building cores, and then it seems to be like a race of how many cores we could get on a single die. I don't know what the biggest one is, but it's big. It's like, I don't know, maybe 80 or something like that. So there's a lot of cores. But for NT itself, that doesn't mean much at all. Not at all. It's mainly the applications.

Saviers: Where do we go from here?

Cutler: So in the kind of chronology here we're at Vista, and we retargeted Longhorn to one codebase. So we're actually back to one codebase. And I'm still an individual contributor, and I worked on Vista, I did some things for Vista. Mainly x64 related, because I've been the guy that was the spearhead for x64. And of course, I say there are four of us that basically did this. We did a lot of it. We did the lion's share of it, but there was a whole other effort going on in the compiler world, where we had a couple guys in the Compiler Group that were just absolutely superb, and they did a great job with the compiler, so there was a whole other group of people. I mean, I don't know if you know how many people actually you would say were responsible for the x64 system, but it was pretty small, maybe a hundred people. Anyway, so Vista, we shipped Vista. And Vista's a big dilemma, because it's like, "Should we ship it or should we start over? It doesn't look very good. It doesn't look the same as XP. It doesn't look the same as anything we've done before, it's different." So we shipped it. And with not very good market reception. Vista was probably the poorest received version of all of the Windows systems that have shipped.

Saviers: Well, there was some other, Bob and some other things.

Cutler: Oh, Windows systems.

Saviers: Yeah, Windows systems.

Cutler: Oh, Bob. Yeah, Bob lasted, I don't know, three to six months. That was about it. So Vista shipped in November of 2006. And I stuck around on the Operating System Group for a while, but then later on in that year, and I think it was around October, cloud computing was becoming a hot topic. And cloud computing is basically moving back the other way where, you know, we had server-- we had timesharing system at one time, then we moved to servers, then we moved to personal computers, and now we're moving back to central storage again, and central computing, where we have a big server farm somewhere that you can somehow, you can interface from whatever your device is, and all the heavy computing or the heavy storage will occur on that end. And we had server farms in several places, but they were specifically for servers for this, or servers for that, or whatever. So there was an incubation project actually that was started, called Azure. And it was to produce a system that would provide a

platform for people to rent, where they could rent compute power, they could rent storage. and we decided it was going to be a virtualized system, because you want to run multiple instances of operating systems, so it had to be virtualized. And this was at a time when the first virtualization extensions had come out for AMD and for Intel, and they're not quite at the level that we thought that we had to have. Specifically, the paging was a real problem. And the virtualization systems that existed-- the virtualization implementations that existed at that time-- they're hypervisors that mainly did shadow pages, shadow page tables. And we thought that direct translation was better. So we decided that we would go with direct translation, which adds more translation levels, but it's direct and you don't have this duplicate copy of page tables on the side. And AMD was the first company that was producing those. Intel came along later. And we decided that our hypervisor would be based on that, and I became responsible for the hypervisor for Azure. And we brought Azure up, and we started a server farm in Quincy, Washington. I don't remember how many servers we had, but it was quite a few, tens of millions of dollars' worth of servers, which was just our test system, right? And we carried on for about two years, and our main differentiation at that point was-- and what we were actually asked to do was produce a system that was a platform as a service, which included all the things that you have to do if you have a server. You have problems with configuration control. Where you have to keep track of the operating system you're running, what version it is, what versions of all the apps are and all that. And then if you have multiple of these, you have to update them all together. So we built all the infrastructure to do all that stuff thinking that that was going to be the big thing that people really wanted. And to a certain extent it is, because you don't want to do this manually. You don't want to manually have, say, "Ah, gees, I've got 150, I've got 2,000, I've got 10,000 servers, and a new update came out for the operating system. Well, I've got to update every single one of those, right?" Do I want to sit there and do that? Well, no. And there were some internal people in Microsoft that actually had developed some tools to do that. And they were mainly the ones who were running our web services. So we decided to produce another set of tools, and I guess, about-- I think about two-and-a-half years into the project, we were doing it long enough that Steve Ballmer decided that, "Well, gees, we ought to really make this a product now." So we made Azure a product. And I forget exactly when the first-- I think 2010 was when we introduced Azure, it was a platform as a service. And at that point, we're out of incubation, and we're worried about, you have multiple data centers, because you want triple redundant logic on all of the storage. And of course, we want to do this internationally, so we're building data centers in Germany-- or in Europe. We're building data centers in the East, in the Far East. Then you have problems about we can't migrate any-- you can't replicate any data across geopolitical regions.

Saviers: Costs money to move it to Canada. <laughs>

Cutler: Well, mainly because of the geopolitical problems of--

Saviers: Right, right, interesting.

Cutler: You know, if you're storing a replication of data from Germany that belongs to the German government, they really don't want it stored on servers in Texas, right? So we shipped the first version of Azure. And there was a change in the management in the Server Group, and a guy that would run the Server Group took over Azure. And at that time, a couple of us decided that we'd go work on the Xbox. And the irony of working on the Xbox is I once sent Steve Ballmer a piece of mail that said, "Steve, I got

an idea. Why don't we take that million dollars a week that we're losing on Xbox, and take out in the parking lot on Friday night and have a bonfire, and roast hotdogs and marshmallows? We can take the people that were working on that and put them on things that'll really produce some good stuff, and we'll save all that money!" Of course, it was just kind of a tongue-in-cheek thing. So here I am like--

Saviers: Going over to the dark side?

Cutler: -- some years later I'm over on the dark side. So the Xbox was kind of a real new thing for me, because this is working on something where the security is something that's much different than the security for anything else. You think of security, the way you generally think about it is physical-- you physically have the machine. It's yours, it's secure. And now you just worry about keeping all those other people out that could invade it through the web or whatever. The Xbox is entirely different than that. We want to guard against people that have a soldering iron and a logic analyzer. They can't break the system. If they do something to modify the system, it will immediately see that it's been modified, and it will do what is called "bricking.' It will brick itself. It now becomes a brick. It's no longer an Xbox, it's a brick. So building such a system is kind of interesting. And it requires a lot of detail and attention to security. A lot of hardware. Not necessarily in terms of cost, but a lot of hardware, a lot of encryption. AMD is in the business of building custom SOCs. And they built the custom SOC for this, and it has eight cores, and they're split into four two-core groups. But it also has a ARM security processor. So the secure boot is actually secure. You can't decide that you're going to solder some parts on. You can't decide you'll take some parts off and reprogram them and put them back. It will discover this, everything is encrypted. It runs with a hypervisor so there's a hypervisor level of protection in the system. It runs-- it's a three-headed operating system where it actually runs three virtual machines, one is called the host system, which communicates with the hypervisor and is part of the trusted computer base. And the hypervisor actually, you could think of that, it's running on the bare hardware, but under the auspices of the security processor. You know, the security processor has the capability to just gate off certain pieces of hardware so you can't even see them. So if the hardware is only going to be accessible by the security processor, it has the ability to do that. It has the ability to just stop the processor. It has a lot of control over what's going on.

Saviers: So back up a bit and say, "Why?"

Cutler: Okay, so the question that we were discussing was, "Why all this security on the Xbox, and make it so that it's-- so the owner can't steal?" And the answer is, "Piracy." The Xbox games are called Triple-A rated games, and Triple-A rated games means that they're very responsive, they're high framerate games. Typically, the people that produce them spend a lot of money producing them. And they're not like a PC game, where typically in a PC game, there's piracy, but they get a big surge when they put them out, and I guess, that they figure that that offsets whatever piracy they get. But a Triple-A rated game is one that's high expense, it might be a hundred bucks. And we guarantee that people can't pirate those games. And we actually get money for that. So if they can't pirate games and we can convince the people that write the games they can't pirate the games, we make money because of that.

Saviers: Okay. So you're talking about all the security functions and you worked on the hypervisor.

Cutler: I worked on the hypervisor, and you know, the hypervisor isolates all the memory. Does all the scheduling. One of the things on the Xbox is that, I mean, generally an operating system you want to share. On the Xbox you don't necessarily always want to share. And there's a lot of actual physical partitioning of the system, depending on what you're doing. If you're running a Triple-A rated games, we'll take a set of cores and dedicate those to the game in a VM. So they don't deal now with the interference that they get with other things going on in the operating system. And there's a lot of housekeeping stuff that goes on. And the other things that go on, Xbox is you actually can be running a game, and viewing LiveTV at the same time, or even Skyping at the same time, or running Internet Explorer. And not having them interfere with each other. So if you're running a game, and it's in the foreground, you got six cores dedicated to you. If you're running a game and it's in the background-- in other words, you're focusing more on watching TV, then you have four cores, and the other environment has four cores. So we have eight cores total. So we're constantly switching these cores around. And the hypervisor's doing all that at the command of the host system. The host system is keeping track of what's going on. So think of this as like you think of virtualization as you're running multiple operating systems. Or you could think of the Xbox as just virtualization. But it's really not, because all this is the operating system, which we call Hydra. And it really has these three different OSs running, or three different VMs running, all right? And one of them, the system OS is called-- is our shared resources environment where you actually could be running things that were like you'd be running LiveTV, or Netflix or Hulu or whatever there, and it has all the UI. So everything you do with your game controller is interfacing with something in that particular operating system. But all the device drivers and everything are in the host OS. So there's some communication between two to three through shared buffers, and the game is sitting in here in another VM, all right, with its own resources, and whatever. So the memory is separate. It's not like a demand page system where-- like a timesharing system where you steal pages from Joe to give to Sally, right? They get what they get, so that every time that game runs, it runs the same. The audio doesn't glitch, the video doesn't glitch. And of course behind this is some pretty good GPU hardware.

Saviers: So this gets you fully engaged in terms of the future of Xbox? Or you have your eye on something else?

Cutler: Right now I don't have my eye on anything else. I'm sort of just trying to keep up with what we're planning to do in the future. I mean, the most I can say about that at this time is it's evolutionary. There are some thoughts about what do we do? Should we really have a console, or should we really try to beef up the PC environment to be able to run these Triple-A rated games, what would that mean? It's not going to be as secure. For instance, on Xbox, there's a bunch of certificates that you get when you buy one that say what you can do. Well, if you're a consumer, you get an Xbox that you can play games on, and you can do these other things that there are apps for. You can't run kernel debuggers, you can't do any of that stuff. So there's no opportunity for people to run those kind of attacks. But on the PC operating system, you can't rule that out, because somebody buys a PC, they say, "I own the PC. It's mine." Now thankfully from the day we introduced Xbox until now, the people that buy Xboxes, they actually have become accustomed to the fact that that environment doesn't really belong to them to tinker with. They can't load device drivers, they can't put code in the operating system. The typical attacks that somebody would have through the Internet Explorer or the browser are not possible. Or they're severely mitigated by the way the layering in the system. One of the things that the hypervisor does is, and we do all this

through-- I won't call it cryptography, but it's secure hashing-- is we use nested paging, which means that the hypervisor has the real page table, and the real access bits. And the VM has what it thinks are the page tables, and what it thinks the access bits are, but it can't get an execute bit turned on, unless the hypervisor turned it on. And they hypervisor will only turn it on if it's in a page that has a hash that we recognize. So you can't inject code in the operating system. If you find a buffer overflow, then the best you can do is maybe you can jump somewhere where you can find a gadget. And a gadget is a bunch of bytes that are executable but not necessarily on the instruction boundaries that do some weird thing. You know, since the x86 is a variable length instruction machine, there's no nice crisp instruction boundaries, so you can jump in the middle of instructions. So that's typically what these guys do is they find a buffer overflow. They jump into the middle of one of these gadgets, which then jumps somewhere else in the code that they've loaded or something. But you can't do that on the Xbox.

Saviers: So let's shift gears into prognostications about the future. What's the future of operating systems?

Cutler: That's a good question.. And VMS is still alive after whatever its 35 years. A little over 35 years.

Saviers: I think it's 37.

Cutler: Yeah, 37 years. VMS, I do not believe is nearly as big as Windows is. Windows, you know, I hate to guess how many lines of code are in Windows, but I think it's well over a billion. Might be several billion. It's huge. Linux is smaller. UNIX is sort of gone. I really thought that Linux would go the same way that UNIX went, where everybody took UNIX, got a license to it, and then tried to add their value by making it incompatible with all the other versions of UNIX, which is great for us. Because it just made it so that there was fragmentation. Linux seems to have not gone that way. I mean, they've pretty much seemed to have held themselves to be compatible. But still, that's a huge system, too. It's not as huge as Windows, because it doesn't have 28 years behind it, yet. Now it's got 15, or whatever it's got. But in ten more years, it'll have that, too. There's a lot of little things that you can do, like there's people that are building little operating systems for engine control, or something. But to build a big operating system-- and to build some level of compatibility to run all these apps that exist is a very large effort. It may be an effort that even the US government wouldn't want to take on. So, the thought that there's going to be a new operating system at the level of Linux or of even UNIX or even NT, is kind of like, it's going to be difficult. I don't think it's going to happen very-- if that happens it's going to be a long time from now. And it'll happen-- if it happens, it'll happen because the complexity, the codebases grow to the point where it's like, "Do we want to solve the complexities of the codebase, or do we just want to throw it away and start over?" And people may say, "Well, we just want to throw away and start over." However, throwing it away and starting over doesn't mean that you can just junk the compatibility, because you can't. You can't junk the compatibility.

Saviers: Is Big Data a fork in the road for what an operating system is?

Cutler: No, Big Data, the way I understand Big Data is just it's employing applications to visualize, analyze, sort through big sets of data to find relationships that you otherwise couldn't find. It's not really an operating system, per se, issue. It's more of a data analysis thing. And it's, you know, it's-- we have these

huge data sets. We were doing this stuff at DuPont. We were doing-- one of the things they did at DuPont, they did their own program called RAMR, and RAMR stood for, oh, Regression Analysis and Multiple Repression, or something, which was, basically, all the time at these chemical plants, you're trying to take the data that you collect and see what environmental variables, human invariables, operational variables, chemical purity and whatever, what correlates with what, so that you can rise to the optimum on some curve.

Saviers: On some process?

Cutler: On some process, right? And so that's part of the Engineering Services Department had-- was a Math and Stat Group that did just that, right? So we've been doing this stuff, but now people are starting to look at it in terms of things like marketing, you know, where you target ads, and a lot of different things. And I don't think it's, per se, an operating system thing. I mean, at some point we may have-- in fact we probably already do have big cloud applications that you can buy and run and provide your data.

Saviers: You have thousands of processors working on it.

Cutler: Yeah, well, one of the things that's nice about the cloud is the elasticity. Is that you can say, "I only want one processor right now, one core." Or you can say, "I want 10,000." And you could get 10,000 processors all running in parallel. And maybe they would be running as multiple instances of VMs or they could be running as multiprocessing within a VM, right? So.

Saviers: Is there any hope for the general decomposition of programs into parallel processing?

Cutler: Well, you know, people keep banging away at that stuff and it keeps getting better. People have been doing that, you know, Ken Kennedy at Rice, Ken Kennedy was the big guy for the Dusty Deck FORTRAN, and infamous. And I'm sure he must be a fellow. Is he not a <CHM> Fellow?

Saviers: I don't know, no.

Cutler: He was part of the Tera/Cray startup. He was doing Dusty Deck FORTRAN in 1970? Where they're taking FORTRAN and trying to take Dusty Deck FORTRAN and compile it so it could be vectorized and done in parallel. There's just lots of stuff going on. I think more of the stuff is going on now to not necessarily take the old stuff and make it run in parallel, but to always produce the new stuff to run in parallel. One of the things in Windows 10, in fact I think it was in Windows 8, too, is user mode scheduling to make it so that you can have tens of thousands of threads, and you schedule them yourself, so that there's a minimum overhead, because you know how you're switching. You don't necessarily have ten thousand processors, you might only have 20 processors. But you have ten thousand things that you might schedule. So there's sort of a move in a direction that's partly was driving that was the core vs frequency thing, where people realized, I don't know when, that was about five years ago or so we realized that-- when-- in fact, Intel said, "Within ten years, we're going to have a ten gigahertz processor." And then they said, "Oops, we're not probably ever going to have a ten gigahertz processor, because of this problem, unless we find some other way to build transistors." So being able to run multiple threads and schedule them yourself became something that, at least, our people thought was essential to being

able to exploit parallelism inherently in programs from the start, rather than saying, "Oh, we're going to take this old program, and make it run in parallel."

Saviers: What do you think of AI? Current state of AI?

Cutler: Well, you know, AI is doing some things now that I think-- you know**,** when AI started there was this huge uproar about, "Oh, wow, gees, you know, it's not going to be long, because the computer's going to be doing everything!" And it wasn't long before we thought, "Well, the computers are never going to do anything!" But now there's a lot of AI that's going on that's pretty groundbreaking. Like well, I would call this AI, I don't know if you'd call it AI, but I think the Skype translator is absolutely unbelievably something that is just unbelievable. This is where-- I don't know if you know about this or not? This is where you can be French and I can be Spanish, and we can actually talk to each other, and Skype will automatically translate it-- when I talk, translate my language to your language. That's unbelievable! That's unbelievable.

Saviers: How many PhD theses were written about this before it happened? <laughs>

Cutler: A lot. A lot. And you know, now we have on Windows we have fingerprint recognition for a logon. And Debbie's like, she just got a new phone-- she dropped her phone, I guess yesterday morning, and broke it. And she says, and she got her new phone and it's got Windows 10 on it. And she says, "Look at this! I can logon with my eye! My eye print. My iris print can logon." So I think AI has really taken off. You know, the other thing, I guess AI, I guess you should think about computer controlled cars as AI. And you know, I think that that's great. But I see on one of these things that you gave me there was a question about, "Will you ever sit in a backseat of a AI car?" And I think the answer to that is, "Not for a long time." <laughter> I tell you why I believe that. I believe that, because that kind of AI is a little different than the speech thing. It's like the difference between having a machine, like a CAT scan kind of machine, where you have a real-time control of that machine, and if that control goes wacky, it kills somebody! As opposed to our speaking back and forth and we garble a word. That's a very different thing, right? So knowing how big and how complicated Windows is, and how complicated and big Linux is, I think that there's a high probability that there is a large number of bugs in the software. Unless it's extremely simple. Now they may have taken Linux and stripped it down-- or whatever they took-- and stripped it down to such a small thing that they can actually reasonably be able to prove that it's correct. However, you know, computers are not 100 percent. You know, they have, every once in a while there's like, "What happened? It just went away." You know, like, well, Tandem built systems that were triple redundant logic or whatever. So I don't know. I haven't read anything about whether they have triple redundant controllers in these systems that control cars, but I would expect they do. And the other thing is, you know, like what happens when-- you know, one time we're in Lancaster, California, and we're going to a racetrack, and we've rented a car from Hertz that has a NeverLost. And guess what?! We lost the NeverLost! And we lost it because the atmospheric conditions were such that the GPS could not work. So what happens when you're whizzing down the highway, and those atmospheric conditions occur, and the GPS no longer works. Your proximity things work, but you don't know where you're going anymore. So I think there's some questions that need a lot of answering before we put cars on a highway at 70 miles an hour, and say, "Yeah, here, we'll get in the backseat and watch TV."

END OF INTERVIEW

