Oral History of Blaine Garst

Interviewed by:
Hsu, Hansen

Recorded July 25, 2016
Mountain View, CA

CHM Reference number: X7853.2017

Hsu: It is July 25, 2016. I'm here with Blaine Garst, formerly of Bell Labs, NeXT and Apple. Let's start from the beginning with your life story. When and where were you born?

Garst: I was born on the left bank of Iowa, <laughs> the town called Council Bluffs across the river from Omaha where Warren Buffett still lives, and his-- he had his mother at a resting home where one of my aunts ended up as well; it was kind of interesting. He's a mega-billionaire but he's just kind of this ordinary kind of Midwestern kind of guy and I kind of view myself as a Midwestern kind of guy, grown-- grew up with very much independence and values and your handshake is all you need, honesty and integrity. My parents got married instead of going to college and my dad took off to the war shortly after—

Hsu: World War II?

Garst: World War II, shortly after my eldest sister was born, and came back and had another-- had my middle sister in '49 and I showed up in 1956 so there's quite a gap there but my parents were older. My dad was kind of a pillar of the-- tried to be a pillar of the community though. He was in insurance. He was dean or deacon at the church I was at and my mother ran the local PTA; in fact she ran the regional PTA. So they were both kind of go-getter types but they lacked formal education so education was very important to them, and of my 20-some cousins I think only 3 or 4 of us ever got to college and so education was not part of our culture. I lived in the city for a little bit and I went out to the country. I had a pony as a child. We had wild animals around and I'd chase my sister around with a snake to hear her scream and stuff like that so joking and pranksters was big, but I always had a knack for knowledge so there's a picture of me walking up the steps with my nose in a book and the book was "How Things Work." I needed to know everything about everything and I've had that kind of insatiable curiosity all my life, so I read science, Science News, Scientific American and Harper's and a few other interesting rags. So I was born there and when I was in junior high we moved to suburbia, to Champaign-Urbana, Illinois.

Hsu: You were born in Council Bluffs. You had moved to the countryside--

Garst: Outside of Council Bluffs. Yeah. All my cousins were in town but-- so we moved out. One of the things I remember was I was in Cub Scouts and Boy Scouts and I learned how relays worked, how you do Morse code signaling stuff, and I go "Relay-- a relay, interesting." Well, a relay if it's on can enable power to another relay and I go "Hmm." I figured out tick-tack toe around that time and so I laid out on paper a perfect solver that would never lose for tick-tack-toe using relays probably when I was-- I don't know-- ten or eleven. And so I was always trying to invent things and taking notes and stuff so I had aptitude and that was known but I just had a fun time riding my bicycle around, throwing dirt clods around and just being an ordinary kid for a while.

Hsu: Were you unusual among your peer group?

Garst: So yeah. I had an unusual name. I went by my middle name, Blaine, 'cause of my dad-- 'cause I'm a Junior and my dad went by Gerald and Blaine was not a name anybody had ever hard of before so I got a lot of flak for that in the fairly small grade school, 30 kids in the class total. I got flak from that but eventually they figured me out and-- but in junior high I got placed in the advanced class there and it was clear-- became clear that the smart kids were not appreciated by the other kids as much. And when I moved to Illinois we went to-- I went to the local junior high for a couple years. I buddied up with one of the other smart kids and-- but he and I-- the teachers would-- they graded us on a curve but he and I were-- always had nearly perfect scores and so the teacher would just ignore our two scores so I'd end up with 106 on a 100-point test and things like that but I learned some things. One of my teachers there basically said-- he introduced the idea of an open-book test-- this was a geometry class as I remember and we were like an open-book class. He goes "I'm not here to teach you exact things; I'm here to teach you how to learn and to learn means you go to wherever you need to and so having an open book doesn't really-- I'm not asking you to memorize everything in the book. I'm really trying to teach you how to learn." And so I think he did that and I thought that was interesting. I also in junior high had a interesting experience. They folded a few of the classes together and they had a block teaching thing for a few hours in the morning and so it was-- this was like 1970, '71 and we had to write a project so I decided then that I wanted to study the effects of having a guaranteed A and my requirement was I write a paper about it. And I go "I can write a paper" so in effect what I was studying was what is your motivation-- what is your real motivation; if grades are not your motivation, what is one's motivation. And grades were easy for me so getting a guaranteed A really wasn't asking a lot but it turns out it gave me incredible freedom and so I ended up kind of being-- helping manage this-- manage-- editor of this little weekly thing we got together. We got all the other kids to write stories and we figured out the production processes and stuff and one of my friends was a cartoonist and he wrote a little cartoon panel at the back of the-- back of our little newspaper and it chronicled the four of us, our attempts to overthrow the student council and we succeeded. We got rid of four out of five members of the student council across the semester and one of those panels-- actually the-- one of the last panels landed us in the principal's office interestingly enough but I'll leave the details of that lost to history at the moment.

Hsu: So you were kind of I won't say prankster but a little bit of an iconoclast?

Garst: I had a cackle that was known so I've been irrepressible-- incorrigible and irrepressible I think for a long time.

Hsu: Earlier you mentioned Midwestern values and your parents. Were your parents religious? Did they have any particular political leanings?

Garst: So we never talked politics. Mom voted one way and Dad voted the other so they never had a conversation about it. I was raised a Baptist. When we moved to Illinois though they only had a Southern Baptist instead of a Northern Baptist and Southern Baptist just didn't really work. I questioned my faith, the faith that was being taught to me, and I went through baptism and-- but after that I kind of had-- I said,

"You know, hmm"-- I had a conversation with God. I signed off with God sort of. I said, "God, I'm not really finding answers and I've asked and I'm not finding the answers that I'm looking for so I'm not going to say that I don't-- that I reject this but I'm going to say if you ever need me you send me a sign and until then I'll just sign off." And so for the rest of my life I've looked for signs <inaudible>. There have been some cases where there are-- things I can't explain have happened. In fact, there is many-- there are several times when I probably should be dead and I'm not <laughs> so who knows? I feel like I've always had a-- some kind of a vision-- I have had a vision. I've had a vision since my Bell Labs days and I've been working on that and so I've felt driven. I have ideas. I know how to make them happen and I feel like I need to make them happen. My mother gave me good counsel. She said, "Just because you're smart doesn't mean you're better and you owe back to the rest of us [the] benefits of that intelligence because it's rare," and that's a message I now go back and teach at my high school on occasion 'cause after junior high I transferred in for two years at a-- it's a public high school, a University Laboratory High School, Uni high school in Champaign-Urbana, Illinois. It's a very special school and it's why I- I'm still affiliated with it and I went there for two years and it was remarkable and so I now go back and teach morals and thinking and technology on an annual basis a little bit.

Hsu: That's a good transition so let's move into that. Your family moved to Urbana-Champaign and how did you get enrolled at this special high school that was an experimental--

Garst: So—

Hsu: --part of the university?

Garst: So the high school has been running almost a hundred years now and they only graduate thirty or forty kids a year so that's three to four thousand graduates lifetime for the school, and three of them have Nobel Laureates and a Pulitzer Prize and in my junior year as I left for college I got the math and science award. So yeah, it was a very magical place run more like a university, very tolerant, very open-ended, and nearly decrepit facilities that we loved. What doesn't kill you makes you stronger; they're very much into that but they're about killed off so I have thrown some money at them and they've built a new collaborative classroom out of that and got some other people to throw in some more money and they're going to remodel the science room and build out a few more things. And I've been told my— [long pause]

Garst: I've been told that it is my commitment and my early money that has helped make all of that possible so I've done some things. I try to do a lot of things to help the world be a better place because just because I'm smarter doesn't mean I'm better and I really believe that and so I need to give back in ways-- all the ways I can. And so I view a lot of my work, a lot of my professional achievements or almost all of the open source and I think I have advanced the industry forward in at least two significant ways, maybe more, and I'm not done. I have a ton of better ideas that I'm working on; that's the last stage of-- this part. So that high school: So I transferred and my mom signed me up for this high school. I mean I was trying to get into physics class in my sophomore year and they wouldn't hear of it, the high school I

was going to, and so she found a high school where I could learn-- take physics in my sophomore year and that was the University Lab High School. I went in there for an interview and they taught different languages. None of them were Spanish so I had to sit in on a German class and there's eight kids in the class and so it's not like I'm not noticed but I'm visiting so everybody's reading from German and so I just kind of pick it up quickly and I read the German out to them and surprised the teacher 'cause apparently I had a gift for languages, which I turned into computer languages later, but-- so I got in. But they didn't have-- they didn't teach Spanish so I had to choose a new language, so I chose Russian. I figured that Russian, given the Cold War, it's better to know your enemy kind of thing and I learned a lot about the Russian people versus the Russian government, but also one of my teachers basically said, "The Russian people are used to strong-arm rulers and this is how they think about it" and that was also a deep thing that I'm-- I've done more recent thinking on as well. To learn the Russian alphabet since it's Cyrillic and not Latin alphabet I got to use -- type on a keyboard-- a computer keyboard, a workstation in 1971. It was the PLATO system, big, huge box. I came to the PLATO anniversary thing a year or so ago but I learned the Cyrillic alphabet. It was a bitmap plasma display. They invented the plasma display for this terminal and then later it became TVs but I used it 'cause it could paint Cyrillic alphabet so it was used as a teaching tool so they had a lesson plan. They had software called Tutor Language that you could-- teachers could write lesson plans to, Q&A questions and other kinds of stuff in there, and so I learned the Russian alphabet on it. Later on I would play Interactive Dogfight, that little keyboard and I was flying an airplane and shooting people. That was a little later, probably 1974, but I played Interactive Dogfight on a plasma display tube in 1974. We had e-mail, we had chat, and so I got exposed to that from my high school because the government-- the national government-- the federal government had invested that much into figuring out how can we use computers in a classroom to give better education and I saw the future; I really did. And so between then and now I've had this perspective of knowing how-- I've just been watching the cost curve of electronics. People talk about Moore's Law. I look at the cost curve of that because just as-- if you know how to build something faster you also know how to make something slower cheaper and as those cheaper devices moved out into PCs for example, I mean I watched PCs happen, eight-bit PCs, sixteen-bit PCs, but I watched that whole trend line so the [i]Phone, the [Apple] Watch. I'm now thinking deep-- have thought deep thoughts about the Internet of Things.

Hsu: I want to get more into PLATO but back up a little bit. You mentioned your mother was the one who saw your potential and transferred you to this high school. Could you talk a little bit more about her and how encouraging she was of you and your learning and--

Garst: Patricia Mary Murphy was my mother, a very Irish name.

Hsu: Were both your parents Irish or--

Garst: No. So I'll presumably talk about my dad as well I assume but my mother was very much a literary type and a force. As I said, she was a regional PTA—

Hsu: Did she work?

Garst: No, she didn't. In Illinois she started having some health issues and she went everywhere, the Mayo Clinic, everywhere to try and find out what was going on with her and it turns out she had celiac sprue and it was not known back then. In fact, she was one of the first people diagnosed with it and when she finally found out what it was and what she had to do, cut gluten out of your diet, it turns out people didn't know how to do that. I mean you could buy out of Portland, Oregon, a gluten-free bread for ten dollars a loaf and have it shipped to you and it was terrible. So my mother was a good cook and she ended up writing up a cookbook, gluten-free cooking, and sent it out to people and people would write her back saying, "This saved our lives" because they just didn't know how to cook. You couldn't go out to eat, you couldn't cook, and so it was a big deal. And so she founded the Midwestern Celiac Sprue Association. That's my heritage. My dad sold insurance. He did a lot of things. He flipped burgers when they first got married, went off to war, came back.

Hsu: He didn't take advantage of the G.I. Bill?

Garst: He didn't. He had a daughter-- I don't know-- had to make money. I never asked him about that-- he-- but he shifted into insurance. One of the first tasks they gave him was to go sell life insurance. So life insurance there's two kinds: whole life insurance is the one where you actually build up a bank account more or less and it was sort of tax deferred so it was a genuine savings vehicle at least at the times. And they gave him a task of going out and selling it in a very poor area of south Omaha and he did that and he would go and sell this and at that time that neighborhood was mostly African American and my dad's very pale-faced like me but he went and sold and he came from that-- came back from that, that and his—visiting Nagasaki after the bomb.

Hsu: How early?

Garst: A few weeks after. He was on a destroyer outside Japan when it was bombed. He saw kamikazes coming down on his carrier group and stuff. So he was in the war and I had several uncles in the war; one of them was in D-Day. So that era is very real to me because I had relatives in it and there was a bunch of combat shows and stuff but World War II, still a phenomenal thing I think most of whose lessons I think we've nearly forgotten at this point and it's a shame. But my dad when he was in Iowa he had-- his face was up on this billboard, "Call Jerry Garst"-- and our phone number was there-- "for your insurance needs" but he kind of took the idea that he was helping these people out and he was. He was giving them security after retirement because his father suffered during the Great Depression as-- my mother's father didn't; he had-- the Murphys were-- he was-- Charlie Murphy was in the railroad business and so they fed the rest of the neighborhood on occasion including my dad's family because my grandfather had been in the real estate business and that just went belly up. And they had some tough times and people pulled together and that's another part of that Midwestern values thing. I think I've figured it all out and normally I wear T-shirts because they're good enough. My dad lived in kind of an okay place in the last years of his

life even though he had money to live elsewhere because it was good enough. The idea of something being good enough being good enough is a good idea and it's not what we are pitched today by consume, exploit, consume, be the best, king of the mountain, you are the most special person, the whole "me" focus of culture in our economy honestly. So he had-- by his lifestyle gave out-- gave a different-- gave different values there.

Hsu: Did you have any other siblings besides your sister or--

Garst: Well, my two sisters. One was 16 years older and the other 7 years older and so they were-- my middle sister was obviously in the same household for years but we weren't close because of the age difference.

Hsu: What was it like growing up during the Cold War and how did that affect you?

Garst: Offutt Air Force Base was sort of Strategic Air Command's-- one of the major strategic air command sites and so it was absolutely one of the targeted sites for a Russian and a Chinese nuke; it was 20, 30 miles away and we- we'd be history in the early phases of any war. And the idea of mutually assured destruction was an interesting game theory problem from my perspective, how do you get out of this kind of situation, and I-- we still have that issue. Nuclear disarmament I think is ever more important but erasing income inequity is actually the core to making that happen when you-- but <laughs> that's the future, not the past so we'll stay away from that one for now.

Hsu: But were you already thinking in terms of game theory even back then in junior high, high school?

Garst: When I was out in the country we would sit out at night and look up at the stars and watch the satellites going by and we watched the Apollo landing that summer and so space and the future were cool and such potential, and looking at the Cold War and the dilemma the planet was in was hard. The riots in the '60s in Chicago, Detroit were harsh, the assassinations, so the world was precarious back then but there was the youth movement and there was a lot of energy, right, and we started seeing-- so it was a time of balance and the Arab oil embargo in '73 also was a wake-up call so we had these national culture shocks back then. It was hard. It was hard to know up from down but the communists were definitely the big evil and yet the people behind them weren't, so I had a more nuanced view of that.

Hsu: Were you relatively insulated from all those things going on either in the countryside or in suburbia?

Garst: Every night on TV they have a body count, a hundred Americans killed, a hundred and fifty Americans killed this week in Vietnam; you couldn't escape that part of it. My sister, one of her high-school friends went off to Vietnam and didn't come back; I remember going to that funeral. So I was

actually eligible for the draft; I was in the draft for one year the last year it was in play so I could have gone. So war and political stability were very much on people's minds.

Hsu: What was your view on the draft and on the war?

Garst: Well, I thought the war was-- I mean I understood it from both perspectives or what I thought were both perspectives and so I wasn't-- I just thought the whole thing was a mess; it was-- it wasn't clear what we should have done. And given how it turned out, it's not clear but it's hard to say what would have happened had we not gone in there eight years earlier. I don't know what eight years of American lives did but it could have been worse I guess; I don't know. I'm not a fan of war.

Hsu: Were you ever into the counterculture or--

Garst: Not at that time and I-- when in-- at Uni no, I was a straight-laced kid. The class ahead of me, a couple kids got kicked out for drugs and stuff-- one kid, but it was more of a prankster kind of a environment at the high school so-- and filled with geniuses, okay, and they got three or four applicants for every one they let in, still; I mean it's the place to go for high school. And so one of the kids in the class ahead of me decided-- he read up on some stuff. "The Anarchist Cookbook" was actually-- and the whole "Steal This Book," Abbie Hoffman kind of thing. There was a lot of counterculture going on 'cause our high school was on campus. So one of the kids decided to have some fun in chemistry and so one Friday afternoon he was in the chem lab and he mixed up a concoction of Tri-Iodine, something like iodine trioxide, something like that, fairly easy to make, and so he made up this solution and he carried it down and put it in his locker and went home for the weekend. And so the drips that he had made on the stairs coming down, the drama teacher kind of noticed when she was walking down the steps because her feet were kind of going "poof, poof, poof," and then the locker exploded and the high-school principal at the time was like "Oh, my God, what's going on?" And so this was before much of what we-- I mean and so they called in the-- they called up the university. They go "We may have some bomb materials in here. What do we do?" and so there was a huge fun-- a huge to-do over little kids-- so I think he got suspended for a few days.

Hsu: Okay.

Garst: When we went back for our high-school reunions they forgot to change the master key that one of my classmates had and so we just kind of let ourselves into the high school and leave messages on the chalkboards and stuff, I mean good fun. I mean we were there the next day and stuff so it wasn't terrible but I mean it was a fun place, very creative, and I ran out of science classes kind of early; I was only there two years. The second year they didn't have any math for me so they invented New Math there, which is-- you have to look that one up, had to do with vectors and other kinds-- it was a big rage through the '70s except it didn't stick so they tried to invent some new stuff there; it didn't stick.

Hsu: A new way of teaching math?

Garst: Yeah, yeah, yeah. Look up New Math.

Hsu: It was actually called New Math.

Garst: It was called New Math and are teaching kids about vectors and dot products and stuff like that.

Hsu: Some matrices.

Garst: Yeah, a different way to think about it they thought, but the-- so I ended up taking chemistry and advanced science at the same time, which included advanced chemistry, but-- so I took two science courses in my junior year and my chem-- for chemistry class this underclassman approached me, Louise Allen, and says, "Hey, I'd like you to be my chemistry partner" and I go "Okay." And so she and I were chem partners and she was also-- I-- but she wasn't-- she also-- she wasn't in the advanced class that I was, but anyway so they crafted a little-- custom little schedule for me to figure out and she and I won the math and science award at the end. The reason I say that is she went off to be a lawyer. My girlfriend at the time graduated summa cum laude from Harvard Law but my chem partner who won this math and science award went off to be a lawyer and so at that time we could be anything we wanted. I didn't know whether I wanted to be a lawyer, to be a scientist, be-- the world was our oyster, it really was, and so what I ended up doing in college was a little different because of that and I think that that helped me along my career as well.

Hsu: Was your introduction to PLATO part of an official class or was that a supplemental thing?

Garst: My introduction to PLATO was because of my class in Russian-- because I was taking a Russian language class.

Hsu: Oh.

Garst: Okay?

Hsu: They deployed it specifically to teach Russian--

Garst: So they had three or four of these terminals in one of the labs up in the third floor and you'd go up there and put on headphones and go through the thing and learn the alphabet. They used it for other

courses as well. I only remember it for Russian but one of my colleague-- one of my colleagues-- one of my classmates got really into it when he also went to the University of Illinois and he got in and talked to the systems programmers and so I had greater access to it. The digital computer lab for the University of Illinois was right across the street from our high school and so we were right there next to it anyway and so he got me more into the deeper ends of it while I was going to college and studying computer science and so I still hung out with him and some friends and we got after-hours access to do those little games and stuff.

Hsu: Was that only when you went to college or were you already starting to play around with computers in high school?

Garst: No, there was no playing around with computers in high school. I mean you could buy a car for $3500, a house for 25,000, and this terminal was about $8000 and it needed a CDC something—

Hsu: 6600?

Garst: No. They had a mainframe and then they had these little concentrator units that would help-- that would talk to the terminals and then talk to CDC so—

Hsu: Was it a timesharing system?

Garst: Yeah, absolutely, and so they coopted it for these terminals so you didn't have access to computers in high school in those days.

Hsu: What was the experience of using PLATO like?

Garst: It was remarkable. It was like "Wow." I mean they had several games and we had Moon Fight, you had Moon Lander, and the gaming was actually the most interesting part of it because the ability to teach and to go through-- you had automated Q&A forms, right? You know, what's the right answer for this? You could do things for math; they had a whole bunch of math stuff, so you solved this equation and it would graph it and stuff, yeah, 1971, '72--

Hsu: Yeah.

Garst: Pretty cool. So the ability was absolutely there but I mean I'd been watching "Star Trek," "Superman" and all these other things where computers were known-- thought to be the saving technology that's upcoming or can be turned into robots and take-- help us all out, "The Jetsons." This

was their image of how computers and technology were going to go forward and I wanted to be a part of it so that was-- it inspired me for sure.

Hsu: Were you big into science fiction at the time?

Garst: No-- well, yeah.

Hsu: Or it was more the moon landings and space race--

Garst: Yeah. I read all the Ian Fleming books and some James Michener and a bunch of other stuff but yeah, I started reading science fiction around then, more in college and stuff, but the early science fiction was Asimov-- was the classics, Asimov and Heinlein and stuff. It wasn't until grad school that I discovered Larry Niven, but I actually recommend to the high school a book, "Diamond Age" by Neal Stephenson.

Hsu: You mean more recently.

Garst: Yeah, to my-- when I go back that's one of the books I recommend for them to read because I still think it's got some very prescient ideas in it. Programming for me didn't start until I skipped my senior year in high school and went to college at the age of 17.

Hsu: Why did you graduate early? Were you just that far ahead; they ran out of classes for you to take?

Garst: Well, again I think the guiding hand of my mother kind of came in there. She found out about this early admissions program and yeah, I mean I'd kind of run out of stuff for them to teach me and they really didn't have a go to college or partial classes in this so there were three of us from our class that went off on early admissions that year. My friend, Armen, his father, a professor at the University of Illinois, solved the four-color theorem.

Hsu: Wow. So--

Garst: So three-- the three of us-- three of us left the class early and went into college early. I don't know how it came about, whether they sent a letter to the parents, whether-- but it was something they were just starting up.

Hsu: And that was something that was special to the University of Illinois because--

Garst: Yeah. It was an early-admissions program sponsor but-- so the university was running the high school as well. It was run by the Department of Education so we had professors who were-- or teachers at the high school who were professors at the university, studying-- introducing new curricula and teaching methods and stuff. And so nowadays I go back and I meet with Department of Education folks. I met with the guy in charge of the genomics institute [Carl R. Woese Institute for Genomic Biology] there founded by the guy who found the third branch of life way back in the '70s. I've had some great conversations with professors back there so I am-- I've reached out to them a bit and it's been fun, but anyway-- so-- yeah-- so again we got university professors teaching us high-school kids and stuff so it was kind of a positive atmosphere and good.

Hsu: When you started college how did you decide on your major?

Garst: Well, being a math and-- being a science guy I decided I was going to be a scientist when I went into college but I'd taken this little personality profile kind of test and—

Hsu: A Myers-Briggs kind of thing?

Garst: Not that one. I'd taken that one later but no, it was sort of-- it was more like you have the attitudes of people in this industry, kind of thing and lawyer was not one of the high points on there. They had something like-- I can't remember the-- they-- what they called it. I mean they called it something-- information engin— information technol-- it was like IT-- and I was big on the IT thing and all I knew was it had something to do with computers and so on a lark I decided I'd sign up in my first year-- my first semester or-- a computer science class; ah, it'd be fun; why not, try it out. So calculus and computer science and a required English class and a few other things but computer science was one of them.

Hsu: They already had a computer science degree already?

Garst: No, they did not. So computer science was taught out of the department of electrical engineering, but they had-- the courses were labeled "computer science" so I took CS-101, so I punched cards. I learned FORTRAN, WATFOR. I was talking to some of my colleagues on the C committee [recently] and they could tell me-- they told me more about WATFOR than I'd known at the time, but anyway so yeah, I programmed in FORTRAN and it was just like solving puzzles; it was a lot of fun and so I signed up for-- I liked the computer-- I liked it and so I-- the next class I took was a survey class in computer languages and so I learned about PL/I, I learned about LISP and RPG and more interestingly both SIMULA and there was another—

Hsu: ALGOL?

Garst: I don't think they actually taught-- I'm not sure. I don't remember whether ALGOL was one of those classes or not but-- so I learned five languages and just went "Oh, this is really cool" so I decided I'd sign up for another class and so my third semester in college I had tested out of-- I took these placement tests going into college and placed out of almost-- well, a semester and a half worth of classes so I went into college a year and a half into college officially already and so that's kind of how I ended up graduating early is I got a bunch of early admissions credits even going in, so I got three hours of credit for college for psychology for example, and we hadn't had a psychology class. It was just… Seemed to know the right answers.

<laughter>

Garst: So the third class, 201, I didn't know it at the time but it was sort of the do or die class, the Department of Engineering--

Hsu: A weed-out?

Garst: --had set up.

Hsu: Where they would weed you out or…

Garst: Yeah. It was the weed-out class. And so in the first weeks they taught us the Pascal programming language. And our first assignment was to emulate-- first assignment was to emulate these, you know, eight instructions of a PDP-11. So they had to teach us what a PDP-11's instructions were, you know. And how to do that. So we were, you know, but we had to allocate registers and memory and they had some test things, but so… And then like the third project was to write an assembler.

Hsu: Wow.

Garst: So you could take assembly language and turn it into machine code. And so they more or less taught us how, you know, low-level machine code, Pascal, everything else, and by about halfway through the class, the class of 90 had dropped down to about 45. And we lost even more people <laughs> by the end of it. I was having fun. And I got an A in that class. And I was like looking at all my other, you know, double-E hardcore students dropping out and I was going, "Maybe this is a thing. Maybe I should think about this."

<laughter>

Garst: You know, "It's fun and okay, maybe I--" you know. I don't know. I did lousy in my first chemistry class. At college. It was more or less a repeat of the class I'd just had in high school. Quantitative Analysis. But I just was not good in the lab. And I said, you know, "I'm not going to be a hard scientist. I'm more a theoretical kind of guy." But they weren't teaching theory, and so I decided to specialize more in computer science. But they didn't have a Computer Science degree. And the double-E guys were going to make me learn horrible things about-- I didn't care about. And so I decided, I shopped around, and I said, "Well, economics is an interesting degree, because--" and I teach this at my high school. For three weeks we had a substitute-- well, the head of the department came in and taught our social studies class.

Hsu: In high school?

Garst: In high school. And she came in, Dr. Ella Leppert.

Hsu: The head of the Economics Department at--

Garst: Head of--

Hsu: --Illinois?

Garst: No. She was running the program at the high school.

Hsu: Oh, okay.

Garst: But she came in and she kind of threw out the program to date and she says, "Why do we have history? What motivates us?" And, you know, these open-ended questions.

Hsu: Wow.

Garst: And she was one of these teachers who you fear and love, you know. She, you know, there was very little slack in the way she assessed you. You knew you were being measured whenever you talked with her, but she taught us that economics was the driving force behind wars. Politics. Technology development. You know, and economics is the study of insatiable-- an insatiable demand meets finite resources. And that was a deep lesson for me. Three weeks' worth of that, and it just stuck home, because I wanted to know how were-- because I wanted to be rich and not have to work, you know.

Hsu: <laughs>

Garst: And so, you know, that guaranteed A from junior high. I was already working on that. I was actually-- it was my plan for when I was retired.

Hsu: <laughs>

Garst: You know. Because everybody ought to-- and I still believe this. And I actually, I really do believe this. I think people do best when they're having fun. I've always had fun. And well, there've been some stretches that weren't fun, but it's been my motivator. So I shifted into economics. And so I started studying comparative economics and… Which taught me about centralized economies and how they can actually provide better infrastructure for the population than merely consumer-driven stuff. And I learned a bunch of stuff about economics. And… But I took more computer science classes than I did economics.

Hsu: <laughs>

Garst: And so I got to, I graduated early, and went off and got a master's degree.

Hsu: In Computer Science?

Garst: I did.

Hsu: Mm-hm. Also at Urbana-Champaign? Or…

Garst: No. At the University of Illinois, I took, you know, I took a number of classes. Operating systems and languages and stuff. I got a job. I was a consultant, which meant sitting over at DCL, Digital Computer Lab, and that, and people would walk in, not students, but grad students and professors, would come in with their printouts and say, "There's a bug here. Can you help me find it?" <laughs> And this is, like, my first semester or second semester in college. About that time when I shift-- I guess that was probably my third or fourth semester when I got that job. And then this one guy came in. I can still picture him. And he put down this printout in front of me and he goes, "Ah, there's a bug here." And I looked at it and it was like, "What language is this written in?" He goes, "This is SNOBOL." And I go, "I've never heard of SNOBOL. You probably ought to go somewhere else." He goes, "Just… Let me explain it." I got, "Whoa okay. So what's going on here?" Well, Smalltalk [SNOBOL] is a context-sensitive language, it turns out. And which is a no-no in the language design for years now, but they learned through that. And so you get a little pattern, you set up a little pattern here, and it's a rewrite rule

here. And so the rewrite rule can create a pattern that exists over there, and so that's why it's a context-sensitive…

Hsu: SNOBOL or…

Garst: SNOBOL is.

Hsu: Okay.

Garst: Language.

Hsu: Okay.

Garst: And so anyway, this guy had this program. So he's explaining this mind-boggling language with, you know, semicolon, you know, on False. Labels. Okay. It's got labels and go-tos in the wrong places and stuff. And, you know, "Where's the data?" And, "Oh, okay." And so we're talking through his program, we get to about page three and he goes, "And then this thing goes-- oh."

<laughter>

Garst: "Thank you." You know. Grabs his printout and runs out. So that made my day. I was able to help a guy solve a problem in a language I'd never seen before he walked in. And so I said I was good with languages.

Hsu: Hm. So did you take that job because it was just fun or you needed the money or it's something to do?

Garst: Something to do. I was taking 18 hours of class, had a full-time girlfriend. I'd moved into campus. I was partying hard.

Hsu: <laughs>

Garst: Yeah, I had extra time, so I took the job.

Hsu: <laughs> Okay.

Garst: Yeah.

Hsu: So you mentioned-- oh, that's a little bit later. So you went to grad school at UCLA?

Garst: Before I leave University of Illinois--

Hsu: Okay.

Garst: --I remember taking a logic programming class. From Professor Duncan Lawrie, who later became a dean of the Computer Science department there. And so he was talking, you know, NAND gate, AND gate, Karnaugh maps, you know, how you lay out logic circuits and stuff. And so they had a little lab that went with it. And I just remember, it's either on a, I think it was a lab test. I think it was a test. You know, they gave us-- they had these, these parts. You could actually build, you know. They had-- you could plug them together and it was better than wire wrap, but you could actually build these circuits in the lab and stuff and… But we mostly, we spent the time designing them. And so on this test there was, you know, "How many parts--" you know, "How do you solve this logic problem, given this number of parts?" Seventeen parts and stuff. And so, you know, I looked at it and I figured out how to do it in 16 parts. By inverting the input at the very beginning and I can share that gate somewhere else and stuff. And so I did it with one fewer part, you know. And the grad student's kind of going, "Oh."

<laughter>

Garst: So, you know, I had, you know, it was a puzzle. You know, it was a puzzle. So that was fun.

Hsu: <laughs> Did you take any social science or humanities classes in college?

Garst: So my degree was in economics. And so I remember taking-- well, I took a film class, so I learned about all kinds of great film directors. I took a class in billiards.

Hsu: Billiards? <laughs>

Garst: Yes.

Hsu: Okay.

Garst: These were P.E. requirements that I got to fulfill.

Hsu: Oh.

<laughter>

Garst: And I took racquetball, which was more of a P.E. requirement and stuff. But I took English, I took a course on Shakespeare. And I was taught by a professor from India.

Hsu: Wow.

Garst: Who had a very poor-- had British accent but not well practiced. And so he was kind of hard listen-- he had a very different expectation for his students. In fact, he gave us only one test across the whole semester other than the final. And he flunked most of us on that first test. And so we walked, we went through this class having class discussions and this, just having no idea how this thing was going to turn out until we had the final test. And the final test was, you know, "Choose one of these three things to talk about." And so one of the things you want to talk about was, was there, in "King Lear," was there any, do you see any parallels between King Lear and Christianity? You know, can, you know-- you underlined the premise. You know, can we look at King Lear as a Christ figure, you know, given all this and that? And so it was like, "Whoa. I never thought about that before," you know. "Oh, mm, mm." You know. So I got inspired. So I-- <laughs> you know, I wrote down this answer, you know, and stuff. And I ended up with an A in the class. But that was like one of those, whoo, terrifying things. For somebody trying to have a great grade point average, you know, you can't manage your grade when you only have flunked one test going into it.

Hsu: Right. <laughs> So how would you sort of--

Garst: And I took a Women in the Labor Market class also.

Hsu: Oh, okay.

Garst: It's not quite social studies, but it was talking about social problems, right?

Hsu: Right.

Garst: Because back then it was 70 cents on the dollar. You know, women earning this, the glass ceiling. I mean, so there were huge issues for women then. And so I got a broad spectrum of economic issues affecting, I think, everybody and not just whatever.

Hsu: So how would you summarize your education at Illinois?

Garst: Fast and furious.

Hsu: <laughs>

Garst: I had a great education. And, I mean, I got to twiddle bits. I mean, I learned machine code. I knew how to build a watch. You know, the little three-button watches LC—[D]. I mean, I could've designed that knowing the logic I got out of, from, Duncan Lawrie's class. I felt I knew, you know, I got, I had, great training and I just wanted more. I just wanted more. And so I went off to grad school the next year, to UCLA, and learned theory. Because they didn't have quite the hardware there that they had at the University of Illinois. So I took classes in artificial intelligence and formal methods, formal languages, proof techniques. Very high-level stuff. And it gave me the theoretical background for a lot of the stuff I just sort of had intuitions about. One of the things I got out of that first job at DCL was access to terminals in front of the mainframe. In front of this-- in this case it was a Digital, a POP, a DEC-10.

Hsu: TOPS-10?

Garst: Yeah, running TOPS-10, I think, and… But we had a terminal, so I could code in my things. And in the Pascal book that I'd learned, or that semester, in addition to, like, you know, just having fun in that 201 class, there was a page in that Pascal book that showed me a top-down recursive descent parser for an expression.

Hsu: Okay.

Garst: And it was like, "Oh. That's how computer languages are done." And so I went off and I wrote my own language and most of a compiler.

Hsu: Whoa.

Garst: In my spare time.

<laughter>

Hsu: This is while you were still an undergrad.

Garst: Yeah.

Hsu: <laughs> Okay.

Garst: I was probably 18.

Hsu: Wow. Yeah. That was your sophomore year?

Garst: That would've been my junior year.

Hsu: Junior year.

Garst: I don't know. I don't really remember when. I've got a DEC tape with the code still on it.

Hsu: Oh, wow.

Garst: I was having trouble figuring out activation records. You know, I was getting to the point where I needed to generate activation records and I kind of lost access and, you know, whatever. It didn't really ship, but I was designing my language and compiler for it.

Hsu: Wow.

Garst: But based on looking at a recursive descent example.

Hsu: Okay. Hm.

Garst: And so that's-- I learn. That's how I learn. I learn by doing. I look at a hard problem and I go and I solve it. Then I go study how other people have done it.

Hsu: I see.

Garst: And I have really deep insights as to what the hard parts are.

Hsu: Right. So it's a interesting combination of both the actual practice and then the theory.

Garst: Yeah.

Hsu: So that you learn through the practice first, and then you go and learn the theory and you go, “Oh, I see.” You learned the-- huh.

Garst: Practice guides my appreciation for theory.

Hsu: Right.

Garst: So I’ve done that in, throughout, my career.

Hsu: Right. Was that the reason why you chose to go to UCLA was because that--

Garst: I asked some professors. They said, “No, don’t go here. You need a variety. Go somewhere else.” And so I applied to Berkeley, UCLA, Cornell, a few other places. But I was late for my GRE. I think I partied too much the night before.

Hsu: <laughs>

Garst: And I had a delayed GRE and I missed the deadline to get in for Berkeley somehow, but I made it into UCLA. And so I went to UCLA instead of meeting Bill Joy at Berkeley.

Hsu: <laughs> Okay.

Garst: Met him later.

Hsu: Yeah. We’ll get back to that for sure. <laughs> So… see… How would you summarize your experience at UCLA?

Garst: UCLA was-- California was just amazing. And shirt-sleeve weather.

Hsu: And <inaudible> your first term. Yeah.

Garst: You know. I was, I divide my time at UCLA into two parts. The first part where I was lovesick for my girlfriend who had gone off to France for a year abroad program, and I went off to grad school and I was just kind of, you know, emotionally miserable, just going through the classes and stuff. But I had a cool roommate, and so he and I had some fun. And I went off. I packed up all my goods into my back of my car and put them in a storage unit and went off to France to live for a year with her, and over Christmas vacation. But it didn't work out. And so I came back and finished the second half. And the second half I learned that-- so in the first semester in, at UCLA, I had come up. I had, you know, my advisor and I'd worked with him a little bit, and I came up with a good idea. Modules and modularity were a big idea back then, and I wanted to know, "How do you get two modules that can do the same thing? How do you describe modules that do the same thing?" And so I wrote sort of this module interface kind of language. I called it LINGO, where you could describe what, two things, that do the same thing. And I said, "Well, if they can functionally do the same thing, then they're--" but there still can be variations on it. That's why you're going to have two variations. Like, you know, one might cost money or one might have, need, a database or something like that. And so into the interface specification I put adjunct little-- I put an extension area where you could write more or less an arbitrary LISP script to do interface assessments. Because modules actually have two parts to them, and most people don't think about the second more important part of them. First part is what they present to you. The second part is what they need.

Hsu: I see.

Garst: And if you don't have a good handle on what they need, you know, you can't know which one, you know, to use. You know, so today it's like when you use this library, "Oh, it needs--" So-and-So other libraries, you know, kind of thing. But I tripped on that. And so when I came back from France, I found out that my professor had sold the idea. <laughs>

Hsu: Huh. It was your idea?

Garst: Yeah. Well, I disappeared on him and he had a grad student, that they had this contract with some aerospace company and, you know, they'd use the idea in there. And so he wouldn't let me use that idea for my thesis.

Hsu: Oh, man.

Garst: I was bummed out by that.

Hsu: Yeah, that sucks.

Garst: So I didn't actually finish my thesis for a couple years. I went off to Bell Labs and skipped the thesis. Because I did all the requirements for my masters, didn't have to have a thesis. But he had talked me into making a thesis and then sold the damn idea.

Hsu: Oh, man.

Garst: But that idea turned into-- he said, "I didn't get a lot of money for it." And I was just like, "Okay." Intellectual property. You know.

Hsu: Yeah. Hm.

Garst: But that idea is literally what turned into protocols at…

Hsu: Really? Oh, wow. Okay. That's interesting. So…

Garst: Which got copied into interfaces in Java.

Hsu: Right. Because I was actually, I mean, off, way, farther down, I was going to ask you where--

Garst: <laughs>

Hsu: --protocols came from. But that's amazing. Okay. Wait.

<laughter>

Hsu: Okay. Wow. So it came from this module thing that you were, I mean, so modules were-- yeah.

Garst: So my thesis was describing two different statistical inference packages, SPSS and I can't remember the acronyms at the moment. SAS, I think.

Hsu: Okay.

Garst: But they were both statistical analysis package. And my professor was just adamant that the social scientists of his era were just misusing science. They were deep in the correlation is causation, down that path.

Hsu: Oh, wow.

Garst: And he was just, you know. So he drilled that into me. And so… But the statistical analysis packages would both, you know, calculate a mean and standard deviation and stuff. And so I had to come up with some modular API to both of these systems, such that you could possibly write one program and choose which one of those two subsystems to use. Because there weren't very many examples of many things that do the same thing.

Hsu: Right.

Garst: Okay?

Hsu: Huh.

Garst: If you think about the client-server problems of the '80s, where manufacturers would build their own APIs and sell them to their clients, they'd get lock-in. You know, IBM had done that and all the minicomputer companies later came in and tried to do that. So that you couldn't have one program that ran anywhere else.

Hsu: Right.

Garst: Okay. And the only thing that'd solve that is interfaces. Abstract implementation of, or abstract specification of implementation. And so in the form of Java, it actually did, it solved it. Because you had Java servers spec'ed out. J2EE is written in, in interfaces -- my protocols. I have James Gosling on stage saying he took it from Objective-C. See, and I can tell you how that happened later. But it did. It solved the client server program so you could have, you could write client code on any platform and run. You could have an Oracle server or this server or, you know, a IBM server, et cetera, et cetera. It solved the client server program, so the-- because it's a blueprint. So that came from modules. That was the idea I had when I was like, you know, 20.

Hsu: Huh. So modules, I mean, at the time, I mean, the various Modula languages. You had, I guess, later on Ada, or-- Modules was like sort of a big thing at the time?

Garst: Well, there were languages. There was Modula. Modula had come out, I think, I think it was little bit-- I didn't know about Modula.

Hsu: I see.

Garst: I think it was in the research community. I don't think I knew about it at that time. But it was objects or modules. And so objects won out over modules. Modules turned into objects via Parnas', David Parnas', information hiding thing, you know. You hide the details behind the modular interface, but it is the implementation.

Hsu: Right.

Garst: I mean, it was that style. It turned into objects, from my perspective.

Hsu: Right. I see. But in some ways they're also kind of orthogonal too, objects and modules. Like, there's-- you can kind of think of objects as being more fine-grained and modules as being larger grained or vice versa?

Garst: So now we look on modules, we've redefined modules over 40 years.

Hsu: I see.

Garst: Modules back then were, modular programming, was the buzz word. And modules were something people were trying to invent. And so I invented a module description language for that, which is more or less an interface.

Hsu: Right. So you say the definition of module and modular programming has changed?

Garst: Mm-hm.

Hsu: Originally it just meant separating the interface from the implementation and then, and that got conflated with objects?

Garst: Objects are the descendent of module--

Hsu: I see.

Garst: I view-- they also were being explored, but I think modular programming just shifted into object-oriented programming.

Hsu: I see.

Garst: And that's why we don't talk about it too much now.

Hsu: Right. But now modules are more like a larger package management system?

Garst: Yeah. Not they're more… Yeah. They have a more specific definition now.

Hsu: I see.

Garst: They didn't have libraries very much back then. I mean, I was at Bell Labs when we were inventing shared libraries. Most code was, you packed it into a computer, you loaded it into the main processor and it ran. You didn't have libraries. Libraries are more a minicomputer phenomena. And so, you know, the '80s, you know, so '70s. So yeah. So that was the best of what I did at UCLA, I think. And--

Hsu: Mm-hm. You could've gotten a Ph.D. but then your professor sold your idea <laughs> and you couldn't finish your thesis?

Garst: No. I mean, I wasn't, we didn't, I mean, I was from middle-class. You know, my parents paid for four years of college education. I had no idea. I had no idea what a Ph.D. would do for me. I had no idea. And I was getting kind of bored with school. I wanted to go out and do something real. So I got-- so UCLA ran [a] nice little recruiting thing. So I interviewed with 16 people on campus and 8 of them liked me enough to fly me around and interview me. And four of them were nice enough to offer me a job. One of which was Bell Laboratories. Another of which was Data General. Data General was recruiting for their North Carolina, for the new, for The Next Big Thing. So Data General invested a ton of effort in this brand new processor and whole new system and didn't succeed.

Hsu: Was that the Eclipse?

Garst: I can't remember at the time. But what I do remember is there-- and a book was written about it.

Hsu: "Soul of a New Machine."

Garst: Thank you. "Soul of a New Machine" was talk-- Tracy somebody.

Hsu: Tracy Kidder.

Garst: Yeah, that's what I was thinking, Tracy Kidder.

Hsu: Yeah.

Garst: Wrote about how, you know, the backwoods guys back in New Hampshire or whatever actually, you know, saved the day for them, because it didn't pan out, if I recall correctly. And what I have to say to that is, well, they failed because I didn't go there.

<laughter>

Hsu: Okay. So what were the other two places that you were recruited to?

Garst: I think Merck, Merck out of New Jersey. In fact, I--

Hsu: The pharmaceutical company?

Garst: Pharmaceuticals. They were trying to figure out how to build factories using computers. So they had a huge IT department trying to keep all their manufacturing and inventory and stuff under control, and they had a ton of money. But they just didn't have a very… Bell Labs offer-- I interviewed four departments at Bell Labs.

Hsu: Oh, wow.

Garst: I had to choose among four departments as well.

Hsu: Wow.

Garst: So one of them was an economic study group. So somebody offered me a job in economics at Bell Labs. A couple were-- I can't remember what the other two were, but I ended up working for a group that was trying to, was called the Bell Data Network. The Bell system was trying to invent the internet. That was the project I chose. The other company that offered me a job also offered me two jobs. It was, I believe it was, ECL. And they were doing-- I went to an interview with them. They were defense contractor type, and basically they could tell me nothing.

Hsu: Wow.

Garst: And they would just kind of hint, "Well, we do business. We build systems for the government. Things that listen to things." Okay. And one guy had a little Pueblo pin on him. Pueblo was a spy ship captured by the North Koreans for about three months. And so these are the guys who kind of built the equipment that went on the spy ships and stuff. And they offered me a hardware job. Or a software job. Either one. I go, "I haven't really done hardware." They go, "You'll do okay."

<laughter>

Garst: I said I'd go do, I would work, I'd watch Ma Bell invent the internet. I figured that'd be a big thing. That'd be useful. So I went to work on that project.

Hsu: I see. Okay. So the internet, not exactly the same as the internet now, but what was, what were they doing?

Garst: Klein was-- not Klein. Kleinrock.

Hsu: Len Kleinrock?

Garst: Yeah. Was at UCLA trying to do, you know, basic networking and stuff.

Hsu: Oh. Yeah.

Garst: Just down the hallway. There were rooms I couldn't go into because they were also trying to secure--

Hsu: Oh.

Garst: Doing some secure stuff as well. There was just a whole bunch--

Hsu: Right.

Garst: --IP stuff going on when I was in grad school and I was just, I was just taking classes.

Hsu: Right. Because UCLA was one of the first ARPANET nodes. Yeah.

Garst: When I was at the University of Illinois though, in that killer class, the S201, one of our TAs was Greg Chesson. He was on loan from Bell Labs research, and he came out and brought some, he bought some tapes. And so we were running on a PDP-11 emulator. I think PDP-11/70 emulator running on a PDP-11/45 or some such bizarre combo. But I met Greg Chesson out there. And, you know, there were-- I glanced across UNIX then, at Illinois. But it wasn't available to students. And out at UCLA though, it was. And so I had my first exposure to UNIX out there. And it was Berkeley's UNIX. And so it had a little bit different command set than what I ended up finding at Bell Labs when I went back to the mothership years later. So I got exposed to UNIX fairly early.

Hsu: Right. Huh. What was so compelling about UNIX that, I mean…

Garst: You could type in lowercase.

Hsu: Okay. <laughs> Was that the main thing?

<laughter>

Garst: It was more human.

Hsu: Yeah.

Garst: Oh, yeah.

Hsu: Than what other operating systems had you been using?

Garst: Well, you know, one of the languages I learned was something called JCL, Job Control Language. It still exists. And that was, you know, DD source in-- standard in equals this, standard out equals that. It

was a spec for how to run the computer to do programming. And it was, ah, just horrible, the interface. The Digital, the DEC, interfaces were better, but they had a-- I can't remember. I had more experience with VMS than I did with TOPS-10. So back to University of Illinois. Since I was, you know, in the, working at the labs there, somehow I got tagged to be a go-to guy for a visiting professor, Professor Donald Michie. Donald Michie put up a million dollar prize for the first computer program to beat a human chess master. So he helped fund the whole AI push towards stuff, and I got to know him as well. During a semester, I helped him navigate the University of Illinois, kinds of things. And so that was kind of interesting. So I got to thinking about AI. I did. I thought about it a lot back then. You know, min-max search techniques. And then I went off to UCLA and took a class in it. But I'd already been exposed to it. And I'd already learned that the game of Go was so much harder than chess, and so I decided ignore chess and think about Go instead, honestly. So I went on to UCLA.

Hsu: Interesting.

Garst: And yeah. Those are some of the good highlights.

Hsu: Let's see here. So was UNIX one of the reasons why you decided to end up at, go to Bell Labs or…

Garst: Absolutely.

Hsu: Okay.

Garst: I knew it was the, you know, that's where they invented UNIX. The project I went to, they were trying to build the internet. So they needed somebody to do languages. And my master's degree was in Computer Science with a specialist in Programming Languages and Systems. I look at my career and that's exactly what I've done my entire career. Programming Languages and Systems. But at Bell Labs, they were trying to build 8-bit terminal concentrators. At the time, a terminal was something you connected at 1200 to 9600 baud; 9600 baud would put a PDP-11 to its knees. So generally it was 300 to 1200 baud. You'd share a PDP-11 with about a quarter mips horsepower, about 10 to 30 people. And you would do nroff. You'd do text processing. You'd write your code using a command line editor, but you had it on a 24-by-80 screen. And the screens of the time, the smart screens were, you could download a form to the screen and it would paint the form and you could do all your input and never need ACKs from your mainframe computer for every character you'd type. This was a huge savings in CPU for folks trying to build interactive systems. And so it was a big deal for us, so my job, first job, was to figure out a generic language for writing forms onto these terminals. Think of what we now think of as termcap.

Hsu: Okay.

Garst: The ability to generically describe a terminal. You know, in 80-by-24 terms and go to arbitrary places and paint things. So I was building termcap. Or that was one of my first assignments. On an 8-bit processor. And so then they had PDP-11s and VAXes also in the slew of things they were talking about. I didn't do well. That was a ill-formed project. We had psychologists in our group. We had two psychologists in our group of eight, and they didn't know what they were doing either, but they figured they would hire psychologists because they were doing language design, and language had to deal with people, so they thought psychologists would be good.

Hsu: Hm. <laughs>

Garst: Bell Labs was an interesting place. If I could speak from an economic viewpoint about it. They were a cash cow. They were guaranteed eight percent, they could make eight percent profit on their costs. And so they had a lot of costs. Including Bell Labs. So huge cost. It's a money machine for them that way. So they spent a lot of money on a lot of things. My first success at Bell Labs was when I read through this little manual they had for their 8-bit processor. They had an incompu-- an ICE thing. They had a way, an in-circuit emulator. NO. They had a way of-- you have special machine and you put this board on it and it could stop and start and run a chip, you know, via an external device. [It's] still done a little bit. But I figured out that this little external device, it had a way to do output. You could do a little thing and it'd spit a character out. So I wrote up a printf program. So I gave printf to the kernel guys, so they could use this little in-circuit emulator, little gizmo guy, to do debugging. Because they couldn't debug.

Hsu: Oh, wow.

Garst: <laughs> Imagine debugging a kernel without having a printf statement.

Hsu: Wow.

Garst: So I gave them a printf statement. They liked me.

Hsu: <laughs>

Garst: And so I moved into a tools group and started doing tools.

Hsu: I see.

Garst: I worked with-- and there was a good tools group there. One of the fellows in the tools group was David Ungar. Who is well-known in computer science terms. He and I were friends. And to some degree, I was a motivation for him to, ended up being a motivation for him, to come out to Berkeley and… Berkeley or Stanford to get his Ph.D. Where he then went on in Smalltalk, around Smalltalk topics, and he introduced, became famous, for his generational garbage collection. And then he went on and did more language work, building out the SELF language. And his grad students formed a company called Anamorphic, which was bought by Sun, and they retooled their Smalltalk eng-- or their, yeah, Smalltalk engine-- no. SELF engine into a Smalltalk engine to begin with. And then into a Java engine and then Sun bought them, and that's what forms the HotSpot Just-In-Time compiler that runs most places Java runs. So that was a connection there. So David and I have done language and systems work forever.

Hsu: Wow.

Garst: So David and his colleagues had built quite an interesting little development environment. We had our custom networking equipment from Digital Equipment. So we built our own little kind of mesh network connecting up PDP-11s and VAXes and whatever else was around printers. And so we had a huge tools group that supported about 300 developers working on this, you know, this, the internet. What would've been the internet from Bell Labs' viewpoint.

Hsu: So it was-- you're working on tools for TCP/IP or…

Garst: No. We had our own networking.

Hsu: Oh, was your own home-brewed--

Garst: Home-brew networking.

Hsu: --protocols and…

Garst: We were home-brewed, so I did home-brew networking. We had our home-brew remote file systems. We had our home-brew kernels. I added system calls. I put synchronization into the-- or I pulled synchronization out of the kernel and made it a little bit-- or I pulled multiplex-- anyway. I did networking stuff at the user level and stuff on PDP-11s and VAXes and stuff. So we had a kernel group. So we read through the kernel. All 10,000 lines of it.

Hsu: Wow. <laughs> Okay.

Garst: And I knew UNIX. And when that project wound down, I transferred out of it up to the UNIX group, the System V UNIX group.

Hsu: Right. Wait. So the kernel that you mentioned before was not the UNIX kernel? It was--

Garst: It was a UNIX kernel.

Hsu: Oh, I see.

Garst: You see, UNIX was-- what did I like about UNIX? UNIX was, you could hack it. I mean, you could hack it. It was today's Arduino back then. You could add little-- the memory bus architecture, the unibus, was very open-ended. The whole device register, you know, kind of mapping thing. My colleagues there, you know, we built our own little three-button devices and stuff and would play with it. UNIX was just very malleable. And at the kernel level, could do new things.

Hsu: Hm. So you guys were using--

Garst: An internal version of UNIX.

Hsu: An internal version of UNIX.

Garst: Yep. Out of Murray Hill.

Hsu: Okay. And you were using that to work on the--

Garst: And we'd add our own extensions and use that as the development environment for all the rest of the people building this rest of this project.

Hsu: Right. Okay. Yeah. I mean, how many-- like, there were a number of different internal versions of UNIX at Bell Labs, right?

Garst: Oh, yeah.

Hsu: <laughs>

Garst: Piscataway version and PWB and eventually what turned into the System V stuff.

Hsu: Right. And so the version you were using was…

Garst: On that project was out of Version 5 and 6 and 7 out of Murray Hill research. We'd get--

Hsu: Okay.

Garst: So my boss at the time would get a, we'd get a tape from Murray Hill, every now and then, and he'd load it up and look it over. And so one day he did that, because, I mean, the PDP-11 had a very small address space, 32K-- integer and 32K of data. <laughs>

Hsu: Wow.

Garst: So when we wrote our kernels for that, they were tight. And so Allan got a-- Allan Glassergot a tape from Murray Hill, and he noticed that there was-- something was wrong. You know, the library, you know, the libc library, was bigger than it used to be. And he kind of dug it out and he found out that the GetPassword.o <pronounced "dot-O"> was like way larger than it used to be and it wouldn't-- we'd link a kernel and it wouldn't link. It was too big. And so he says he called up Ken, Ken Thompson. He says, "What's up with this, Ken?"

<laughter>

Garst: And Ken laughed. <laughs> And he told him. <laughs> And you can read about that in Ken Thompson's Turing Award lecture.

Hsu: Okay.

Garst: He'd put a two-level virus into it.

Hsu: Wha-- <laughs> Deliberately?

Garst: Trojan horse.

Hsu: <laughs> Just to screw with…

Garst: Have fun.

Hsu: People. <laughs> Okay.

Garst: So Ken told my boss colleague friend, Allen, the password.

<laughter>

Garst: So there was a version of Bell Labs UNIX that deployed inside the Bell system and on PDP-11s such that I think it was when you-- they would supply the area code. They would actually look up your area code and find the right switch to send your phone call to. And on those versions of PDP-11s deployed throughout the Bell system, Allan could walk up, login as root and type the magic password and it would let him through.

Hsu: Wow.

Garst: So the deal was, he embedded the-- the bug was actually, if I remember correctly, he put a two-level virus--

Hsu: You mean Ken Thompson?

Garst: Ken Thompson put a two-level virus into the assembler. So when the assembler would see itself, it would reintroduce itself. But it would also, when it saw the GetPassword routine, insert a different-- extra code that would check for a magic password, and let that one through as well.

Hsu: So it was like a backdoor that he built in.

Garst: A double back door. So the code-- and then you could compile it and it would erase itself. So you could-- the source code was there-- the code was only in the binary. And so the assembler had extra code in it, but the compiler didn't. The compiler would issue the right sequence of code for Get Password, and the assembler would recognize it, and reintroduce it into, god, into the assembler itself, I believe. When it saw itself as being assembled. You can read the details. But it was a two-level hack, it was pretty wild.

Hsu: Okay, and it was deployed throughout the Bell System so that-- I mean, Bell was using PDP-11 for switching their phone networks?

Garst: Yeah, it was one big Bell System. I mean, we owned Western Electric and the Bell, and all the Baby Bells. So this was all one big happy family.

Hsu: This was all live, wasn't it?

Garst: Yeah, yeah.

Hsu: Wow!

Garst: So there were a million people in the Bell System when I joined it.

Hsu: Wow.

Garst: One organization. A million people.

Hsu: <laughs> What was the general culture or atmosphere at Bell like?

Garst: Did I say cowboy? <laughter> There was a lot of duplication of effort. But they had so much money, they didn't care. So departments would compete. Everybody-- there were many people doing UNIX, there were other people writing their own operating systems, because they didn't believe in UNIX, and they had to-- so there were multiple operating systems--

Hsu: Just within the Labs?

Garst: Within in Bell Labs. Yeah. Development organizations. I'm not even talking research. The development organizations were writing their own operating systems and playing around and stuff.

Hsu: Wow. Wow.

Garst: Yeah, so the "not invented here syndrome," definitely. Bell Labs. They had to invent it here. If they hadn't invented it, it wasn't any good, and we'd just do it from the whole cloth.

Hsu: Wow, hm.

Garst: That's how we thought about things. That's how I learned to think about things.

Hsu: I see. And was that also something that motivated Ken Thompson and Dennis Ritchie to do UNIX in the first place?

Garst: Oh, you can read history on them. I mean, they had the Multics experience and there was-- I haven't actually read up enough on the language history before C came out. You know, BCPL and stuff like that. One thing I dug up the other day-- ALGOL-68 had garbage collection, void, struct and a couple other keywords we use in C today. So there was a lot research going on. They were researchers. And they just wanted to do something cool. And so they went off and had some cool ideas with their colleagues. They're the two names we know of, but the whole department was in on it. You know, pipes-- oh, dammit. I can picture him, but anyway.

Hsu: Oh, yeah, somebody said something about pipes.

Garst: Yeah, pipes came-- pipes, oh, I can't remember the guy. Well, I never met him, but I knew of him. But--

Hsu: Yeah.

Garst: So I came in, and since I worked on System V, I ended up meeting and talking and negotiating at times with the Bell Labs Researchers, who had moved on at that point to Plan 9 for the most part. Unix was old news for them, and they really weren't working too much on it. I helped finish-- Dennis Ritchie had done something called STREAMS. Which was a different way of doing I/O at the low level. And we had adopted that into System V. And I helped finish that off eventually when I moved onto that group. So when the internet project, Bell Data Network, I mean, they spun us off. They spun us off into a startup. It was called American Bell. And they gave us a nice--they printed medals up. And the medals had the AT&T logo, that you recognize today, on it. And on the back of it, it says, "First to be chosen." American Bell. And then the consent decree happened, and they couldn't use Bell in their names. And so they changed the name of it. The project died, and they reabsorbed us, and all kinds of stuff happened. So I got spun out of Bell Labs twice. Also the UNIX System Laboratories. I think I might have been one of their employees for a little bit of time.

Hsu: Okay. And this was all the fallout from the--

Garst: Divestiture and stuff.

Hsu: Okay, the breakup.

Garst: Yeah, part-- the consent decree in '56 didn't allow them to be in computing. And so they were talking about, they were doing communications, and so everything we did kind of had to under the guise of communications. So there was a constraint that we lived with. Anyway.

Hsu: So you were managing the System V team.

Garst: Well, I was a manager on it. It had four-- about four managers for each of some of the subsystems. But I managed the cent-- the integration point. The central subsystem at times. I mean, when the project was winding down, a friend of mine from Bell Labs had left the Labs. He was a supervisor working on Big Iron on IBM-based UNIX. He had moved out to California. He was working at Hewlett Packard. And he flew me out, and we talked about whether or not it'd be a good spot for me to work on networking. Because I'd-- you know, homebrew networking, homebrew kernels. I mean, we were UNIX guys. I mean, he knew we'd do anything. And so he recruited me to come out and join him at Hewlett Packard. And I declined.

Hsu: What year was this?

Garst: That would have been around 1984. That fellow was named-- his name is Tom Jermoluk. And he-- we'll get back to him, I think, later in the post-employment stuff.

Hsu: Okay.

Garst: But so I didn't come out to HP at that time. I transferred up to Summit, New Jersey and worked into the System V group. Where I brought a lot of general UNIX knowledge. They were working on UNIX kernels and so I could-- unlike the supervisors there, for the most part, I was really a hacker. And they were a little bit more manager types, so all the distinguished MTSs up there just loved me. I could talk turkey with them, "What are we going to do? When are we going to do things?" So I enjoyed being a manager there. I challenged one guy, I said, "You know, we got these files systems and they need memory in the kernel. We got networking , and they need memory in the kernel. We got just process—they need memory in the sys--" And I says, "How about instead of having three or four different memory allocators, have copy to memory back and forth, we have one memory allocator, so when you get data in from the network, you just hand it over to the file system right away?" He goes, "That's a good idea!" And so then about a month later, I go in, "How's it going?" He goes, "It's a harder problem than we thought!" <laughter> "But I love it!" You know, and so I liked motivating people. I kind of sized the guy up and figured out, you know, what would be a challenging project for him and he could sweat the details. I could just kind of give him big direction. And he loved it! And so I started out managing--.

Hsu: That was your first management experience?

Garst: No, I was manager-- I was promoted at age 26 at Bell Labs.

Hsu: Okay.

Garst: So I was four or five years into Bell Labs, I was-- became a manager.

Hsu: Oh, because you had managed the Tools Group before.

Garst: I'd managed one of the Support Group, and one of-- yeah.

Hsu: Right.

Garst: Yeah. So I mean, we had, you know, we had entry level. You know, minimum wage workers we'd bring in, and have them run our machines. So when they would crash, we had to go through the FSCK protocol and stuff, it'd be 20 minutes to reboot a machine. So for that project, there were a couple of notable things from that project. It was big enough. They spent a billion dollars on it in late '70s money. That's a lot of money to spend trying to build the internet. Well, didn't work for them. We had so much money, and we were dealing with custom hardware from DEC and stuff like that. They sent us up. I had to sign my first NDA. I got sent up to New Hampshire with a tape, with a mag tape with one of the guys from my group, and we had two hours of access on a breadboard at VAX 11/782, the first dual-processor VAX. And we booted UNIX on it in that two hours we had at it. And it was the first time UNIX booted on a dual-processor, because DEC's team had another, had their own operating system, it was called ULTRIX, they hadn't gotten that far yet. <laughter> So I booted UNIX on a dual-processor, and so back then, I mean, dual-processor? We got two things doing the same thing. How the heck are we going to do that? We had to solve the problem from scratch. And so we came up with our solution, and we went up there and it worked. And so that's the way I'd been taught to think. But I'd think about the hard problems, the stuff that hasn't been solved yet. So that was my first experience in multi-core, multi-processors. Probably around 1983, maybe 1984.

Hsu: And the VAX was still a prototype at that time.

Garst: At that time, they were figuring out how to manufacture it. So that was their breadboard version of it.

Hsu: Huh! And was it only that model that was dual processor, or--

Garst: Well, they became a line. 11/782, I mean, the PDP-11s, or the VAXes, VAX-11/-- no, not 11-- VAX. 780, 782, I can't remember the numbering sequence in there. But it was the dual processor version of the VAX. And so a year later when I went up to the UNIX group, they were working on a dual processor. It was used to run the telephone switches, 3B-20, it was. And so a year or so later, I actually shipped the first version of System V that dealt with multi--dual processors. And so I was in the middle of the kernel, so what do we do with the locks? And how do we do this? And so how do we keep networking and this and that separate, and so I was thick of all the debates as to how we did, you know, shared libraries. They had a static, the System V fixed library scheme versus dynamic shared libraries. And all that little, and kernel plugins, "Should we have a kernel plugin interface or not?" So device driver plugins or not, you know? So this one guy, I remember this guy coming into my office, and they were working on trying to figure out how to do graphics. And so this guy came in and he says, "You know, we should put Postscript. You know, Sun's been doing this stuff with “Postscript." NEWS. Yeah.” I think that-- yeah, I thought-- yeah, we want to put a Postscript interpreter in the kernel. And I kind of looked— [makes a face] And I was one of the, you know, innovator types. That's why he was in my office. And I said, "I don't think so!" But they-- I mean, this was the level of what we were trying to talk about. How do you do networking? How do you do it efficiently? And so I ended up, you know, I ended up managing the general, rather than just the networking group, I ended up managing the central group that did integration. We had our own remote File System Group, Stateful, RFS versus NFS. We still only had one file system support. You couldn't hook up any other type of file system. And so there was a lot of work going on. And so in System V Release 3, I negotiated with Sun to get their Virtual File System Switch, VFS and NFS, I took their code and we slammed into System V. So I negotiated that.

Hsu: Right.

Garst: We got symbolic links out of that one.

Hsu: Right, okay.

Garst: And on a dare, for a bug fix, I got /proc into the kernel. ‘Cause this was a better way to do debugging, I told them. And I was-- had enough stature that that was the way it was. And so I signed my second NDA a year or so after joining their group, in late '84. And this was in '87, I signed another NDA, and I was put on a plane and I flew out to California. And I sat down across the table-- I didn't really know what we were doing. Sat down across the table with Bill Joy, I knew enough what we were doing, and he and I went through the different system calls that SunOS had and System V had. Sun had been doing a good job of pulling in support for System V. They wanted to unify the UNIX market, because everybody had their own version of UNIX, and no client code would run anywhere else. You know, unless it came straight from Bell Labs, and nobody had hacked on it, you know? And this wasn't good for the business environment. So they were adopting System V-isms. And so I had gone out and brought in some SunOS. And so I got tapped to go out and build a unified UNIX. So we decided on STREAMS instead of sockets, for example as the primary interface to networking. And I thought that was pretty interesting. Because I

thought that's what Bill Joy had done was the, you know, all the ARPA stuff. But he agreed to go with streams instead of sockets, and we went down the list. He proposed there was an fchdir(2). You could change directory based on an open file descript. And I kind of looked at that, and go, "What do you use that for? "Blah-blah-blah. Ba-da-blah-blah-blah!" I think they were using it for a source code control system. I'm not sure. I go, "Did Roy do it?" And they go, "Yeah." So I said, "No, we're not going to do that one." Later, I found out that, in fact, I didn't know it, but I had a check behind my head, I had a check for 20 percent of Sun Microsystems, because that's what ended up happening. AT&T Bell Labs bought 20 percent of Sun Microsystems to gain access to the SPARC chip. It was a hardware deal. I was the one to negotiate the software part. <laughter>

Hsu: I see, wow.

Garst: Yeah. So I did. And so that's how I moved out to California.

Hsu: Okay, wow. That's really interesting. There's a lot I want to pull on there. <laughter> Let's go back to the general like, you know, we talked about UNIX, the UNIX world being so--

Garst: Fragmented.

Hsu: Fragmented. I mean, from your position as the System V Manager, what was AT&T's goal for UNIX, and how did, vis-à-vis the other UNIX vendors, and how did sort of that Sun partnership happen?

Garst: Well, so much like Java, Sun seems to have just copy/pasted Bell Labs' mistakes. And they went out of business. Bell Labs is out of business. So they didn't really have a good business model. So, I dealt with some of the business guys, trying to make money at AT&T out of UNIX. And so they would-- they had a whole big copyrighted, registered trademark. There's a registered symbol on my cheat sheet there. Because it was a big deal. You couldn't say UNIX without saying it's an adjective to something else, for trademark reasons. And so they had all these trademark stuff. But you know, the researchers had let the cat out of the bag by letting out these mag tapes with very sparse licensing agreement. And so everybody had gone off and done their own thing. And what are you going to do? You gonna sue them out of existence? That's not going to build your market. So they had a business-- their goal was to build the UNIX business, to standardize it. And so they were publishing something called *The System V Interface Definition*. It looks a lot like a POSIX book. I still have the book. But it would detail the system calls, the return codes and the commands. And it was a guide, it was general guide to UNIX, and every System V licensee, there was a test that went along with it, and they were required to conform to System V. So Sun, they kind of liked that part of it, you know? "We want to standardize UNIX, so we can build a bigger market and defeat Microsoft." It was, you know, and everybody else, Domain OS, and all the other OSes that were out there. So I thought that was a decent deal, and so, yeah. So we designed the monster mega-monolithic-- it was a *horrible* design! The biggest, baddest, hard to debug thing you could imagine.

But it had everything that anybody could want in it. And our redeeming grace was, we had a three-phase plan. That was the first phase. Phase 3 was to replace it with a microkernel. Replace the whole bottom with a microkernel, so you could do all that heavy lifting in separate protection domains and stuff. And so we set up shop out here, a little sort of applied research group, joint Bell Labs, so Bell Labs had a location on Sand Hill Road that I and three other people occupied for a time while we were working with Sun on this Phase 3, this joint collaboration, the new basis, the new micro-kernel UNIX for the entire industry. And the rest of the industry just kind of didn't like that picture, though. So they formed the Open Software Foundation. It was called the Hamilton Group for a while. But the rest of the industry unified against that vision, formed the Open Software Foundation, somewhat legitimized the notion that open source could deal with an operating system. And although their operating system didn't go very far-- I mean, they ended up picking Mach up out of Carnegie Mellon, but by building this monstrosity, which is now known as Solaris. Solaris, it still-- it runs-- is a System V shipping, still ships System V. And I don't know who else does at this stage. But Linux had sprung up. Linux was the real answer to the Open Software Foundation's efforts that--I mean, those kinds of things spawned up, so.

Hsu: Right. So at the time there was the Sun/AT&T alliance, which, and Solaris sprung out of that. And then there was the opposing UNIX OSF--

Garst: There were tons of UNIXes out there. DEC had theirs, they called it Ultrix. Domain Apollo had a different operating system entirely. But oh, I can't name all the peer-- I mean, there were just tons and tons of companies had their own variations on UNIX. Most of which, Sun was re-releasing their own-- they start with SunOS and add their own variation. So Sun had built-- and System V were the two main source points. But there were about eight or ten companies, major companies, on the System V bandwagon, including Silicon Graphics, for example.

Hsu: Right.

Garst: Which I'll come back to later on.

Hsu: Okay, and I mean, you mentioned that by this point the original UNIX people, Thompson and Kernighan and Ritchie had moved on?

Garst: Yeah.

Hsu: Had you had much interaction or contact with them earlier?

Garst: Well, not too much. When I was at Bell Labs, or when I was at Summit, you know, I took over the Networking Group because the supervisor there decided to do something else. But he had already

negotiated and gotten the code from Dennis, and you know, they had it pretty well underway. And so the third-- the Phase 3 part of this joint agreement was about building a new thing. And so I went back and I did talk to-- and we tried recruiting out of Murray Hill. So we talked to several people, and in fact, we got some time from one of the researchers there, Tom Cargill, who was adamantly opposed to multiple inheritance that was going into C++. <laughter> And I absolutely believed him! And he was another one of the impetuses that led to protocols and interfaces. But anyway, I went and met with several of the Murray Hill researchers at the time. I mean, they were all doing other stuff. So I mean, I shook Dennis Ritchie's hand once. I had one interesting discussion with Ken Thompson over Plan 9. And then I talked to some of the other folks about what we were trying to do with microkernels and stuff like that. But they were researchers and that wasn't the research they wanted to do. We were trying to build code that shipped, and they wanted to do research. You know, it's not the same.

Hsu: So let's talk about the microkernel then. So why did the group decide that you wanted to go with a microkernel, and oh, maybe explain first what a microkernel is. <laughs> And is there a difference between that and a nanokernel? What is the difference?

Garst: I always called what I wrote a nanokernel. I don't know that anybody else has used that term, but I used that term 20 years ago. So microkernels were a big research topic. Avie Tevanian was pushing Mach as the microkernel.

Hsu: At CMU at the time.

Garst: At CMU. And I--

Hsu: Was he very well known in the Computer Science community?

Garst: So there were USENIXes. USENIX was sort of the trade show for UNIX. It wasn't really-- it was sort of semi-research but mostly applied. So people would write papers about stuff they were hacking. So it was like applied research. "Oh, we can get this thing. We can make--disk speeds this way. You know, let's look at this file system, look at all this," you know? So USENIX conferences, I don't know whether they're still going on or not. But Avie would get up and tout the benefits of microkernels. And his thing was actually the memory subsystem. He had copy-on-write memory chunks and stuff. And was pushing the memory system onto sort of this microkernel idea that his advisor, Rick Rashid, had already put together. So microkernels were, you know, the size of the kernels and the ability-- engineers like to ship perfect code. So the kernel-- back to your first point-- the kernel is the program, that is the program that runs, that takes care of all the nasty little hardware details and let's your applications run. So your applications negotiate with this first program called a kernel to gain access to get memory and access to your keyboard and to your display and stuff like that. But all that access has to be, is where you also get and keep protections, and other kinds of stuff. Some-- all that heavy lifting is done in the program called the

kernel, and it's just this huge single piece of code. And the bigger your code is, the more likely it has bugs. And so the idea of, how do you get to correct kernels was to make them smaller. Was to partition the tasks into their own little mini-address spaces, and use some kind of IPC, inter-process communication between them to make everything work. And so there were a lot of efforts trying to make that happen. None of them succeeded.

Hsu: I see. So it was like this notion of plugins that you mentioned earlier. Having plugins.

Garst: Well, modules fit into it, but yeah, absolutely you'd have a modular interface. Inside the kernel there had been some object-oriented techniques even in the first UNIXes, so the way you wrote a device driver was, you had a switch table, there was a five-entry switch table, and based on your device number you switch on the device number, you know, to pick which element in this table, which object to use. So it's an array of device driver objects, with five entry points, and so there were techniques, object-oriented techniques. You couldn't reuse implementations unless you hand coded it, but you could. You could—the switch out is the same as this other driver switch out and stuff. And so there were object-oriented techniques inside the kernel, they just didn't know them as such. And so we tried to do the same thing for file systems. So you'd have a file system switch, which is what we called it at Bell Labs before we bought, before I had to go to VFS. And you'd switch out to a very different file system driver, and so that whole switch-out mechanism inside the kernel was well-known and understood. Or to some of us at least. And so we reused that pattern for file systems as well. Yeah. I'm sorry, where we were going on that one? I don't remember what-- I got lost on it.

Hsu: <laughs> So your version of the microkernel, the microkernel or the nanokernel, I guess, that you called it-- why did you call it a nanokernel versus a microkernel?

Garst: Okay, so Bill Joy had been inspired, I believe. I haven't talked with him about it, but he had these ideas for a microkernel that made sense to me. They called them doors. They went on to build an operating system called Spring.

Hsu: You mean Sun did?

Garst: So Bell Labs and Sun partnered for a couple of years trying to build Phase 3 until AT&T lost its stomach and bailed on the joint endeavor, because they'd divided the industry instead of united it around their plan. So they decided to back off. So I got abandoned in California, and went to NeXT as a result of that. But while we were there, the joint, you know, this Phase 3 team came together and tried to design this new nanokernel. What would it be like? And so they hired some researchers and stuff to-- and brought in some of their SunOS heavies. And there were about eight or ten of us total. And we were going to build a nanokernel. And there had been some distributed operating system research done by David Cheriton, the V System, that was very interesting. And one of the more interesting things about it

was-- and the one that Bill was pushing was the idea of a super-fast IPC transfer. And the idea was that most IPC calls, or most subroutine calls, most of the parameters fit in your registers. Even then, with eight registers. And Cheriton had built that into his V System by saying, "Well, for an IPC, what we're going to do is we're going to just push the relevant parameters in the registers, and just directly jump to this different address space, and you don't have to do-- you don't have to teach the kernel how to do memory mapping into that old address space, and do copy-protected copy of memory." You know, so it was-- it saved the kernel's work in extracting the real parameters and sticking them into some code block, and you know, it's just a different way to get in and out of the kernel without having to flush addressing registers and all kinds of stuff. And so there were tremendous savings from going from one process to another, if you only had to just remember your registers. And so he built-- and so that was what was called a door. And what turned into something called Spring. Or the Spring OS. And so it was very fast IPC. And after-- so he had some design stuff, but as part of Phase 3, we brought in like Bertrand Meyer of the Eiffel language fame. And he talked to us about his language. I went out-- I met Bjarne Stroustrup at one of the workshops. And I had an idea for him and he goes, "Oh, that's interesting! Why don't you write a paper on it!" and I got very scared, because I wasn't sure I knew what I was doing. But he seemed to think I did. And if I didn't, he did. And that was a weird combo, so I never wrote that paper. <laughter> But that later became-- was the genesis for Fast Invocations, which turned into blocks, which are closures. So I actually did follow up on that a little bit later. It's still a good idea. Hasn't quite been done yet. So I went off and I wrote my own microkernel, after AT&T decided to split us off. So I wrote my own microkernel, or nanokernel. And it was, again, an operating system designed from scratch. Except I knew operating systems by that point. So I put some things into my little nanokernel that have yet to see the light of day. But I'm not done yet.

Hsu: Interesting.

Garst: I have some more days left. So I posted a challenge to the-- I believe I know how to make even the L-4 microkernel go faster. For example, and they are the fastest IPC on the planet right now, but I believe my prototype was faster than that. And my first implementation of my prototype, I could cross address spaces and come back, at the cost of ten times a procedure call.

Hsu: Hm. Okay.

Garst: That's always, C++ dispatch. You've ten times the cost of the C++ dispatch call. I could get into another address space and do things. And normally it costs 100 to 1,000 times that. So it's a 100-- 10 to 100 times faster. With that level of speed for IPC, you could build a multi-kernel, where you had a file system as a separate process, and networking as separate processes, because when it's only ten times the cost of a procedure call, you can jump back and forth, okay? With traditional IPC's it was a thousand milli[seconds]--it was 100 times more to go from process to process, and it was unfathomable, untenable. Performance sucked when you did the multi-kernel. And so I explained this carefully to Avie Tevanian when I was interviewing for my job at NeXT. I said, "Mach's IPC isn't fast enough." He says, "Oh, really?"

I go, "Yeah." And so we had a little discussion about it. And I came in and interviewed at NeXT. <laughter>

Hsu: We'll definitely get to that later. Well, so by multi-kernel you meant, the traditional microkernel?

Garst: So a microkernel, we have them today almost. These virtual-- these-- when you have virtual-- god, what do we call that stuff now? <laughs> I don't know, Parallels and stuff, when you have--

Hsu: Virtual machines?

Garst: Yeah, virtual machi-- well, when you have the hardware machine. You know, what do you all the virtualization technology, where you're writing an entire operating system, down to the--some version of the operating-- you're doing machine emulation. You know, VMWare and stuff. So they run a very tight kernel that pretends to be other kernels. That's truly a multi-kernel. What I was-- and that was possible, but mainly what I was trying to talk about in a microkernel is that, you have a very low-level small piece of code that runs when--at boot and then spins off a process to do, say, TCP/IP and another one to do DOS file systems, and another one to do UNIX file [system]. And so all of the kernel work--

Hsu: At user space.

Garst: A user space with address protection. So if you have a bug in your file system driver, whatever, it doesn't crash the whole machine. That's the idea. So you get safety through better modularity there. But the interconnection cost between all these little subsystems was too high for it ever to be practical. Unless they had a really fast IPC, which I have.

Hsu: Right, okay. And so the nano in nano really just means fast IPC that solves that other problem.

Garst: The nano means the smallest kernel you can imagine. It's an executive. An executive plus memory management. Turns it into a kernel, a nanokernel. But very little beyond that. Most microkernels are still-- they really don't exist.

Hsu: I see. Hm. So not even Mach? Like how much is--

Garst: Mach has never shipped! I mean Mach 3 was a real microkernel with a couple extra processes. It never really shipped that way. They have Mach inside, but it's a massive single-image kernel. Just like it always has been.

Hsu: Oh.

Garst: There's Mach on Apple products, there's Mach, and NetBSD, and you know, a bunch of other stuff.

Hsu: But Mach in its theoretical full-microkernel version never shipped that way? It shipped as a hybrid.

Garst: Yeah, it never shipped on its own. Because Mach alone doesn't have an idea of a user.

Hsu: Oh.

Garst: Doesn't have an idea of a file system. It's just got IPC and memory, and CPU capabilities and tasks and threads and things like that. But none of the admin that gives you users and protections and stuff like that. That comes from the next layer up.

Hsu: Right.

Garst: So Mach is an incomplete operating system. It's just a kernel.

Hsu: Right, right, okay. [I] want to talk a little bit more about working with Bill Joy. So what was that like, and I'm just--

Garst: Bill's a character. <laughter> Bill's a character. You know, super bright guy. I ran across him a couple of times at-- in Summit, New Jersey, he'd fly out and negotiate with business guys, and you know, I'd see him in the hallway, or shake his hand or something like that. But I came out, I think to the first meeting for this joint project, and they brought in their team, and we brought our team out there. I mean, he strolls in, he's got a-- he takes over a room. You know, he came in with his, I think Jack-in-the-Box burger bag, and you know, was munching out, and talking about how his Ferrari had to go in the shop. It was nine thousand bucks. He had to pull the engine to replace the battery or something like that, you know? He was just dropping-- but then he would just launch into this technology, that technology, "And what should we do? This." And you know, in a room with researchers and whatnot, it was wild open-ended discussions. You know? And he brought a lot of ideas into it, and he brought a drive to it. It was his-- he was-- he had this idea of the fast IPC. He knew it was critical. And so then he would go off, then he would disappear. You wouldn't see him for two or three weeks. So he was inconsistent on that project. And so the researchers eventually went off and rewrote a version of Mach in C++, and called it Spring, and it had a heavyweight IPC to begin with.

Hsu: Mm hm.

Garst: They didn't follow Bill's notion of doing it with doors. And so they, I think, built architectural flaws in there, because you have to build something above a very pure base. So the right way to have built it-- the way I was building it at least, so in my opinion, theirs differed, whatever. Mine hasn't been successful either, so who knows who's right? But my idea is you build it with the small IPCs, and when you have more data than will fit in registers, then you use a special, you use one call that says, "Oh! Go find the extra data over here." So you make it a two-part call to get extra data around. Which turned out in the Spring OS case was 90 percent of the calls were handled by their fast IPC, which they added later. But architecturally, they were always going through fast track. You know, they were going down special case paths to get to the fast IPC, instead of it being the core logic.

Hsu: I see.

Garst: So it became poorly structured, and they didn't know. It was a research topic, and they never shipped it. They never knew how to replace Solaris with it. So it got lost.

Hsu: So their version of Spring was different from the nanokernel that you were working on.

Garst: Yeah, I called mine Fizix.

Hsu: And so that was their--

Garst: And we'd split up.

Hsu: Oh!

Garst: AT&T had pulled the ripcord and said, "You guys aren't working together anymore."

Hsu: Okay, so they went off and did their thing and you--

Garst: They formed Sun Labs. It was Mike Powell and James Mitchell. Oh, Rob Gingrich was in on some of the talks. And I'm sorry, I can't-- I'll think of some of their other players. But they went off and formed Sun Labs, and then became-- which then took on Java eventually as well.

Hsu: Okay, so Java came out of Sun Labs.

Garst: They did a fair amount of-- there was a fair amount of Sun Labs involvement. And again, Java was a kind of the Bill Joy kind of thing. See his name's on the language, right? So you know, so when you're designing a microkernel that's going to become UNIX and solve everything, you have to think hard about all the things you need to do. And so the hardest problem I thought that needed to be solved was that of security, of protection, of how do you delegate security checking? You know, what does it imply, you know? Does my driver just hand off all security checks to somewhere else? Can you have a centralized security check? Can you distribute it? And so I thought long and hard on those problems, but I thought long and hard about security. How do you get a secure bootable system? And when I was at Bell Labs, one of the things we looked at was something called The Orange Book, which was put together by the Defense Department to say, "Mandatory levels of--," you know, so we had mandatory access control and discretionary access control, and all these different kinds of weird ways to look at UNIX from this angle of multi-level protection domains and things like that. So I thought about the protection issues there, and realized the kernel was the agent for doing this, and how can you really get it done? And I knew that these doors, this fast-trap kind of mechanism was effectively an object. It was effectively a file descriptor, except it does other things. And so you just put an object wrapper around that and you can look at this, what turns out to be a capability like an object. And I thought this was just remarkable and awesome! And so I built an interface description language on my nanokernel that basically said, "When you think about another system, think about it as a remote object," more or less. And so that's how I constructed my nanokernel. And it basically had tasks or address domains. And spoiler alert, it had threads that weren't attached to an address space. The threads actually moved. So the kernel didn't keep track of threads as being bound to an address space. Threads could simply move, requiring much less kernel work for them, and all kinds of other stuff. And so there's a couple things that have to happen when threads move and that is the basis for how I believe I can make the L4 microkernel even faster. Threads are an anachronism now, we don't need them anymore. They were put in there to simulate multi-core and we have multi-core, so we should get rid of threads, we'll get to that.

Hsu: Okay. So you were writing the nanokernel in C++?

Garst: It was my only C++ program.

Hsu: Right, so what do you think of C++?

Garst: It's overblown, it's byzantine. It was byzantine when I looked at it 20 years ago, it's not gotten better.

Hsu: Is it because it's overly complicated or?

Garst: Well, it's… the first premise was, it was gonna be a better C, and so in order to be a C, they couldn't change enough about C and so it's got a lot of bad ideas in it from C. C had a lot of good ideas

given the time but from a object perspective, from a software engineering perspective, it doesn't have very many controls.

Hsu: Right, because of the pointer stuff, you can just do whatever?

Garst: Yeah, pointers and stuff, right. So.. but C++ allows you to have-- you know, you stack allocations, you know, you get into pointer… So it's overblown and so almost any sequence of tokens is meaningful and recently they added-- and it is a context-sensitive language it turns out--you know, you parse later to find out what's going on up there and that little bit of context sensitivity turns the preprocessor into a Turing machine. So you can build you know, the halting problem in your header files. Your compiler will never finish and they know this. I'm not a fan of that. <laughs> So I mean, I talk with the C++ guys, you know, all the time. It's just, it's almost a full-time job to keep up with it and it's said that nobody on the planet actually knows the entire language at this point. So it is the experimental language. It's the experimentalist language and the C language is the one that supposedly doesn't change. Except it does change and I've changed it, so that's what I'm doing lately.

Hsu: So what was your relationship like with Bjarne Stroustrup?

Garst: I only met Bjarne once or twice and you know, purely over technical stuff, you know. I mean, I do debates with him sometimes. He comments on some of my C language proposals and I proposed garbage collection. I have an approved proposal to add closures to--as a technical spec for the C language.

Hsu: This is recently?

Garst: Yeah, that was in April. And then I added a keyword to the--I added a keyword to the C11 language called "_Atomic" and they looked at it and they said, "No, we're not gonna add 'atomic,' add it in this, we're doing it our way," which can't be done, can't be used for C. So… you know… they're not being… you know, as they grow, they're growing in ways that are ignoring C and that gives the C committee a bit of a challenge. How do we grow a language if we… you know, so I have ideas.

Hsu: Right… because C++ was supposed to be compatible with C but they're moving in ways that make it incompatible?

Garst: It was never--they started out by breaking enumerations. They were--enumerations were the first thing that wasn't the same and so there's lots of little things that aren't the same. But all the operating system interfaces are written in C. Had they been moved over to C++, then I'm not sure C would exist,

except you can't build libraries out of C++ because for performance, you know, you change a header file and everything has to be recompiled, so it's super fragile and they made… yeah, so, yeah.

Hsu: You mentioned Bertrand Meyer came at some point?

Garst: Yeah.

Hsu: And was that as a contractor or?

Garst: So that was in the Phase Three group with Sun.

Hsu: So Phase Three was?

Garst: Phase Three was--there was--Phase one was the massive multikernel it turned into--or everything kernel that turned into Solaris, and the third phase of that project was to have been the microkernel. And so what language should we use for the microkernel? Modula-2, Modula-3, Eiffel, C++, C?

Hsu: So you're just exploring all the different languages available?

Garst: Yep.

Hsu: And so you invited Bertrand to talk about Eiffel to investigate if that would be?

Garst: Yeah. So Eiffel had one of those elements from my idea fro--it had on his interfaces you could put some assertions and so that was pretty much exactly my idea at age 19 in grad school.

Hsu: The thing that you said ultimately became protocols?

Garst: Well, that work, that idea, one aspect of which became protocols. The aspect that didn't become protocols was that for every element within there you could put a Lisp script to test assertions or whatever. You could have assertions, you could have cost estimates, you could put whatever else you wanted in there. So in Bertrand Meyer's case he put some pre and post-assertion conditions on entry points. Doable. Very doable, very related, very close to what I'd been trying to do also. You look at interface less as a whole--as a collection of pieces and then each piece can have different attributes. Pre and post-conditions was something he added and that was obvious to me at that time but again for me,

what's the cost of this interface? What's its run rate, what does it need, do you have authority to run it? I mean, you can ask all kinds of questions.

Hsu: So this is sort of the design by contract thing that's part of Eiffel?

Garst: Yeah. That's just, those are the contract elements, but similar to stuff. So I mean, that was a, you know, a good idea, so we--but didn't choose that.

Hsu: So any particular reason?

Garst: Didn't get--the project got abandoned before they got anywhere close to code and so I had enough ideas I started writing code after the joint project was abandoned.

Hsu: Okay, I see.

Garst: And again, I got an IPC--IPC for process to process at 10X the cost of--it was 20, but the goal was 10--20 in the first prototype. 20 times the procedure call cost and I think that's still pretty good. I hope to revisit that accomplishment.

Hsu: So you met Avie first at a UNIX conference?

Garst: He gave, you know, some talks and I'm not even sure I met him at the USENIX talks but I had gone to a few of his talks. I was, you know, I mean, I was a manager, I was shipping Unixes… I think at that stage I had wrangled enough or I had tickled enough political points of interest that I had acquired a whole VAX for my group's exclusive use. So we could write our own kernels and debug our own kernels on an experimental machine while we had about 15 others for the rest of the project to actually use. So the idea was, you know, it was hard--getting access--and the VAX at that stage was about a 300,000-dollar device. So the resources in order to actually do experiments in Unix were far and few between and you had to be very careful with what you got done. So we looked at--you know, so Avie was giving all these talks on Mach and this and that and, you know, there was some impressive stuff there in limited areas, but I don't know. I had a, kind of a skeptical attitude towards it and so when we were shutting down the Menlo Park Bell Labs facility, I--but I was impressed by--I mean, Steve J--you know, Avie had moved on to NeXT, to NeXT Computer, Steve Jobs was running it. And I went and I saw a demo of a NeXT, you know, and I fell in love with the software there, and so--

Hsu: What year was that?

Garst: That was 1990. And so I decided that was-- no, I'd seen the demo-- I can't remember. No, I think it was at a USENIX, I think I saw a NeXT the first time and I kinda was impressed by the software there. So when it came time to look for a new job--

Hsu: You weren't interested in moving back to the East Coast?

Garst: They offered to move me back there but no I wasn't. And so yeah, so Avie interviewed me to work in--he was running the kernel group inside Software, and so he went to bat for me. I got interviewed, I still have my interview sheet, you know, who interviewed me. But NeXT had its own style about things and so I think there was-- if I recall correctly there had been some skepticism about me from somebody on the team. I don't know who it was but Avie went to bat and talked to-- twisted some arms and got me hired on and so I started working for Avie and his kernel group to begin with.

Hsu: And was that because you had challenged him in his interview with you?

Garst: Well, did I challenge him? I mean, I don't know.

Hsu: Or you critiqued his Mach kernel?

Garst: I told it like it was, and he saw that. Avie's a sharp guy. We, I mean, there's a certain level of programmer's experience, you know what's going on, you can, you know, you know one when you see one and he saw something in me I guess and so I got brought on and one of the people on my interview team was Bertrand Serlet and I think Bertrand saw something too. 'Cause I had built this little IPC mechanism just--you know, to glue stuff together and Bertrand had built this funny little thing he called remote objects and then glued-- it was kind of an IPC-- it was an IPC system, you know, at the object layer and so that's exactly what I had done too-- so, you know, Bertrand probably put a good word in for me. I've never asked him about that. I should. He probably would know the politics there. Alright, I should ask him. But yeah, so I got brought on to work in a kernel group and it was amazing because they were about ready to ship a release and Avie didn't really give me anything to do. I was like sitting there for like a couple weeks. I got my login, I got my NeXTstation, I got my office and, "What am I supposed to be doing?" It was like-- they didn't-- you know, “I don't know?” So I asked around. "Oh, fix some bugs. You wanna fix bugs?" "Okay, I'll go fix some bugs," so I went and I started fixing some bugs. It was like-- and it was true. My concerns about Mach as a technology were absolutely dead on. It was a research, it was a, you know, a grad-student project. If you tried to do a-- one of the first bugs I fixed was, if you tried to do a select(2) call from a second thread, it would panic the kernel. The select(2) code wasn't thread-safe. So the Unix part, which was sockets, was not happy with the Mach part, which was just tasks and threads and address-- you know. So it's a huge layer of stuff that Mach just doesn't do.

Hsu: They just weren't that compatible?

Garst: They hadn't coded that up yet. They hadn't glued it together. It's a work in progress. I'd been dealing with people writing remote file systems and remote networking and, "Does it work, does it work together?" I mean, these are the problems I solved as integration manager at System V. "Is those code ready?" You know, and I was in charge of how we integrated code into System V and I made one of my guys read every sing-- I mean, we had a merging process and you know, very elaborate. How do you integrate code into a kernel? Because kernels are horrible to debug. So I had pretty decent discipline around that and knew how to do that, so I-- they just hadn't finished this kernel and then I mean, it just wasn't finished, it had bugs. So I fixed a few bugs and then got involved in-- Bertrand came over one day and talked to me about IPC, you know? I said, "Well I know how to make things really fast-- that's why I got hired, I thought." And he goes, "Well, you know, how fast is it, you know, when you're really going across a machine." "Well, across the machine is milliseconds." You can't make-- "Well until this part of the problem becomes paramount, you know, why don't you work on expressing it better-- you know, here's this Objective-C line," which I thought C++ was gonna take over. I thought Objective-C was, you know, gonna be a has been. He says, "You know, in Objective-C we get to express it this way and that way," and I was going, "Yeah, well, okay, what about this and what about that?" And so he and I sat together and we said, well let's put this idea of this remote, of this module specifi-- let's build this new thing called a protocol," and so he and I wrote up a proposal for protocols that says, you know, here's how you capture.. you know, the modular, the interface definition language, in the language itself, straight up. I mean, this was a revelation to me. "My God, you can actually add-- change the language?" I mean, I'd written compilers and stuff but actually designing a language is a new idea for me. So Bertrand goosed me there and said, "We can actually change the language," and so we wrote up a proposal and brought Steve Naroff into it, who was actually manager of the group and he added to the proposal and then we shipped protocols within months of me joining there. About three months after I joined, I cooked protocols up with Bertrand and did that. So I had a dual road. I was-- and so then after protocols the first thing I used them for was Distributed Objects but then protocols were also used for archiving and so protocols were used to help describe all kinds of reusable behavior at the application level. So I kinda bridged the gap. I was sort of the application support language guy and I was the kernel guy and I helped rewrite, like, the low-level kernel debugging protocol, KDB. When you gotta remote debug something, you know, or use this-- well, I had to revise that protocol.. so I worked all over the place.

Hsu: So you were officially in the kernel group but you were working--

Garst: This is a startup. You do what you need to do. Yeah.

Hsu: Right. And so I mean, 1990 you had.. NeXT had already shipped hardware?

Garst: Yeah, they'd shipped their hardware, I think a year before I think. They were just coming out with… what were they just coming out with? They were coming out with 2.0.

Hsu: NeXTSTEP 2.0?

Garst: Yeah, hang on, I'm trying to remember... it's a little fuzzy there but they were also working on the NeXTdimension, the color board because at that point NeXT was just using two-bit graphics and so they were working on the NeXTdimension and that might've been really what the big release was about, I can't quite remember. But you know, they had their agenda, they had their shipping dates and you know, I pitched in on that. And then Bertrand recruited me to start working on Distributed Objects and so I-- to build Distributed Objects, I needed a better memory model. NeXT at that point was using malloc and free at their object layer. You would 'new' something or other and 'new Window,' Window, you send the 'new' message to NSWindow or NXWindow, or I can't remember what we called it then, I think it was NXWindow and you get a window. And when you're done with the window you'd send it 'free' and it would maybe know better, and not free itself, but basically you were telling it, it was done. And it turns out that's a hard way to write an entire application. No string objects, we had C strings. So we had C strings which were all over the place, pointers, there were still tons of pointers and you could build applications, but talk about ten times longer because the memory was just impossible. So you'd build an application, you'd never free anything, 'cause as soon as you freed something it would crash, 'cause it was interrelated with stuff. So to work on Distributed Objects, I introduced a reference counting protocol for the Distributed Objects part and so we actually shipped that further on.

Hsu: Wait, so reference counting was originally a protocol?

Garst: I built it as-- to be a Distributed Object, I think in that very, very first version I think you had to adopt the reference counting protocol.

Hsu: So it hadn't been built into the compiler, it was just a protocol on top of?

Garst: It hadn't been built in--yeah, I mean, it wasn't on NSO[bject]-- I mean, it was for objects that you thought needed to be remote, they needed to do this reference counting.

Hsu: Oh, so it was only in the Distributed Object?

Garst: This is the malloc/free. This is the first Distributed Objects. Before Distributed-- even before it got renamed that. The first shipment of my version of remote objects introduced reference counting, you know, for the constituent parts.

Hsu: Okay. So it was up in the framework layer?

Garst: It was up in the-- yeah, above the kernel.

Hsu: Above the kernel?

Garst: Yeah, so I had an unfortunate tragedy in my life around that time. My wife passed away, age 33, just turned 33 and I had an 18-month old son and Steve reached out to me and offered me… he said, "Whatever you need, let me know," and that was a big deal to me. I mean, when Steve Jobs calls you up and says, "Whatever you need," it's a big deal. He asked me what I was doing and I said, "Well, getting a nanny for my son." He goes, "Well, I have a son about your age, as you know, and I have a nanny for him, that's a good idea. Could our nanny be of any help to you?" So Steve had a heart and from that and other interactions with him, I have to regard him as a friend. I mean, he did the right thing. So I mean, I remember I'd be sitting at the Apple's cafeteria years later, he'd sit down next to me and say, "Hey, what's the time Blaine?" You know, or, "Did you hear about such and such?" I mean, he'd just open up a conversation. You know, I have a video of him walking up on me. I was videoing the campus 'cause I had a 3 CCD video camera. I was going to the Greek Islands for the second time and who was walking down the aisle but Steve Jobs himself so he comes up and we chat about the camera for a while.

So, anyway, I shifted out of managing kernel group stuff into a more lead contributor role and cooked up-- and that's when the Foundation stuff started happening and so I shifted out of kernels and into application support specifically, although, you know, I just didn't have any kernel deliverables anymore. I did things like-- I started working on Objective-C a little bit more, and, for example, multi-threading. Objective-C was multi-threaded 'cause they had multiple threads, except it was bloody slow. Every time you had to dispatch a message you had to take a walk to make sure that the hash table that the runtime used was sane and that was very expensive and so I remember sitting on the black couches in the second floor of Building Two at, on Seaport and having a discussion with the compiler guys a little bit and said, you know, "I read this paper from USENIX by David Black and he had this idea on how to do kernel level synchronization and what he did there in the kernel was, he said that the synchronization code started at this address and ended at that address and so in the kernel if they ever swap out a process and find that the program counter is in that address range, they wrote that address range such that it would restart the PC at the beginning of it." So this a, sort of a transactional memory approach, actually is what they're calling it now and so Intel has actually built some of that technology into their hardware at this stage for doing a different way of synchronization, but anyway that was an interesting idea and I thought about it and I go, "That's what I need to do for a multi-threaded dispatch." So I invented this idea of lockless reads of the cache, of Objective-C hash tables, because dispatch mainly needed reads. So we got rid of the lock for the, for you know, 95 percent of the, of dispatching and that was a big win. That made multi-threaded application code work. In the Foundation work that I went off to work on with Bertrand, he and Ali Ozer and I wrote most of the principal classes with input from the AppKit group and other folks. So he did.. Ali did NSString and that was enough, because it was huge and massive and I did… oh, I don't know, Bertrand and I and Ali came up with this thing called Property Lists which was sort of our version of XML, before XML existed, you know, and JSON and all that stuff. So we came up with Property Lists but they were based on particular classes. So Bertrand and I decided and Ali decided on how we did mutability. We had immutable objects, had abstract, had mutable subclasses and although we kinda knew it wasn't right, it was the best we could come up with, and only now do I know what we should've done and so I have a better solution to that now. So we did the whole Foundation, the architecture of

objects. We had a working AppKit but, you know, we retooled it, we invented a Unicode string instead of the char*s <pronounced ‘Ch-are stars’> and so for the most part Bertrand's vision there was to take the pointers out of-- the C pointers out of Objective-C programming. And so all the heavy lifting could be done with objects and so you could still do C and we went for a time even with an object form of rectangle, but it was too slow and so we went back to structures and stuff. So there was back and forth with the designs, but protocols got adopted more and more and so in Foundation everything came together. We decided to use the reference counting on all objects. We decided to use protocols to describe this archiving to disk back and forth behavior, because I also reused it to archive an object into a remote message that goes to another system. A Distributed Objects part of it. And so the same archiving protocol was reused for at least two or three different-- and we also used it for the Undo Manager. So the Undo Manager used this archiving protocol with a third implementation, right? I got to build my modules finally. I finally had a module description language and I could build my modules and they used them all over the place.

Hsu: And that was the archiving part of it?

Garst: Yeah, but it came from protocols. The idea that you could reuse this one interface. So you'd write your class and only had one entry point for archiving, you didn't know whether it was going across the wire or going to disk or going into an undo manager. It didn't have to, it didn't want to and shouldn't and didn't. So it was a good way to do abstraction.

Hsu: Okay. I wanna back up a lot 'cause a lot I wanna pull on. So you never actually worked with Avie after you got hired?

Garst: Oh, no, he was my boss. Oh, absolutely.

Hsu: But you never worked directly on the kernel itself or?

Garst: Well, I was working on the Mach kernel there, yeah, I was a manager for Mach kernel. In fact, I was leading a Mach 3 effort. We had some great people in the kernel team. Oh my gosh. Brad Taylor had done some great work at Sun. He had done-- he was one of the NFS guys and he was at NeXT and he was doing a compacted encrypted swapper. You know, you could swap-- you could compress bits on a page and write 'em out and delay writing them out and read 'em back in quicker than you could do the disk writes and so we were doing some compression swapper and you know, and we had some great people. One of the other guys there, John Siemens was a roommate I think at Stanford with one of the founders of Sun, Andy Bechtolsheim I believe and John was a really quiet guy. We came in weekend-- came in one Monday morning and he said, "Hey guys, come on over." So we went over to his office and he said, "See, this is a regular program. When you run this program and guess what? It now has profiling output here, and we can run prof on it and figure out what it was doing." Up to that point, you'd have to recompile your program to have profiling callout so it would-- when you entered this routine you'd have to,

you know, tickle the marker thing, but he'd figured out how to do it without recompiling and adding code. He had done it because he had read the 68020 instruction manual and he found out that it had a certain trace mode. If you turn a trace mode bit on, on a control register and the kernel itself would take a trap whenever it branched, then you could do the data logging. And so he'd built this system up over the weekend. That was the kind of environment we were in. Brian Pinkerton was kinda interested in this thing called World Wide Web and so he and his group, the data group, Adam Hertz and Jack Greenfield, you know, were doing all-- they were trying to do databases and objects and all that sorts of stuff and I had opinions on that so I got involved with that but Brian Pinkerton went off and split off and founded-- and started a company called WebCrawler. It was one of the first web search engines. We had some smart people there. <laughs>

Hsu: I mean, what was the NeXT culture like? I mean, you had all these brilliant people?

Garst: So Steve Jobs' real talent was hiring good people. 'Cause he didn't invent most of that stuff but he sold it. He had ideas, and some of his ideas were the inventions and some of them got battered around and thrown away, you know, but he knew how to hire great people. And the people he hired, the reason they were great was they were great together. It was a very egoless environment in all honesty. People had their ideas, they would push 'em but they could be talked out of 'em. They could learn, we were all there, we were small enough. As you said, I was doing kernel work, language work, you know, whatever work needed to be done, we were all in it together. So it was a very positive environment. It was the closest work environment to my high school that I ever had. Tolerant, directed, but you know, and focused.

So Foundation introduced I think scalable applications. I mean, with memory management and with string objects, it actually… critical to the success of NeXT applications and that went for several years. But we actually became viable that way. The Distributed Objects, that whole Foundation layer, including the remote objects part of it, my Distributed Ob.. so I did Distributed Objects and Task and Thread and most of the operating system layer--Task and threaded locks and timers and the run loop and you know, most of the system level core of how the stuff works. Those were the objects that I led the designs on and wrote up and coded up and did. So they're still shipping. <laughs> They made Apple what it is today.

Hsu: Can you talk a little bit more about your relationship with Steve Jobs and like how often did you interact with him at NeXT?

Garst: So I was an engineer and he was Steve Jobs. So we crossed paths honestly very infrequently. I got called in-- Bertrand brought me into a meeting with Steve when we were introducing I think CMU's Kerberos stuff and Bertrand could not tell Steve why, you know, superuser couldn't override a Kerberos thing. You know, security, kernel stuff, so he brought me into a meeting to more or less tell Steve he couldn't have what he wanted. <laughs> That took about 90 seconds. I mean, it didn't take long. Steve was very sharp and he figured it out. "Oh, okay, so this is mathematical, you know, versus just you know,

operating system code decisions," was more or less the basis of-- more or less it. 'Cause a security-- you know, security assurance is a mathematical property really, ultimately, and that's what we're relying on in our-- that the math proves that this is true and that it's not—ok, whereas kernels could be-- have bugs of their own and be compromised, as we all know. There was just a lot of stuff going on there. Steve came into my office one day and he asked me, he says, "I heard about-- somebody told me about-- have you ever heard about this new file system called such and such," and… "Have you heard about it?" And I go, "No." <laughs> And I go, "Where's it out of?" He goes, "Well, it's being done by so and so," and I go, "Hmm, where's he out of?" He goes, "He's out of Carnegie Mellon." And I go, "Well, you know, from a security viewpoint, you know, I'm not really, you know, I don't think it's-- you know, it's probably-- I mean, I don't know that-- and you know, I kinda doubt that it's gonna be very good." And Steve just stopped and he looked at me and he goes… he gave me some advice. He says, "You know, you ought to… you should be careful. Lest you be accused of things that people accuse me of… and that thing that they accuse me of is called arrogance." "Okay." <laughs> Yeah… yeah, so make of that what you will. <laughs> So I went to NeXT so I could learn from Steve.

Hsu: Was he part of the interview team?

Garst: No, he-- well, he called me up some time after my interview with Avie and you know, they're working out the details, the proposal or something and yeah, Steve gave me a call and talked to me about the job and stuff and… yeah, actually he called me while I was still of course employed at Bell Labs and we had one of these fancy PBXs. They had about, you know, 300,000 dollar PBXs for three people in the office, full time, it was a little weird, but I was on the phone with somebody else. I had a line-- "It's Steve Jobs.." "Yeah, let me…" and I hung up on him by mistake 'cause there was too many buttons on it… he called me back. But we talked about-- I talked about stock options. I didn't know anything about stock options at that point but I talked about stock options anyway, you know-- so many stock options, you know, it doesn't seem like a lot. Steve goes, "Well, you know, I've made more millionaires, more millionaires in Silicon Valley than anyone else," and, "You know, it takes time," and this and that, back and forth and so… yeah, I hired on.

Hsu: And did those stock options make you rich?

Garst: Well, no actually. Well, maybe… I mean, it depends on rich. Now this is a very deep subject and I will not be as jovial if I talk about it. So either we can shift moods or we can't. It turns out my stock options from NeXT didn't make me super wealthy. I'm not-- well, whatever… something else.

Hsu: So what was your overall view of Steve Jobs as a person?

Garst: I loved the guy. He… when the iPod came out first couple years, there was high demand and some teenager in Brooklyn I think got murdered over one and Steve found out about it. It hit the papers

and according to the papers, he called the parents up and said the same thing to them that he said to me. He says, "What can I do to help?" He was a passionate guy about good things and he appealed to people who had that kind of mindset and so I am still friends with my NeXT colleagues. Ran into one out bicycling a couple weeks ago, hadn't seen him in years. It's just instant camaraderie and I go to these NeXT gatherings, it's just camaraderie, it was a special time. We were doing stuff and it was being ignored. I can tell you stories, I should tell you these stories. We tried our damndest to build a profitable product and at some point the hardware we didn't-- you know, couldn't make it on Motorola hardware, we shifted to Intel to try to be an Intel-based NeXT computer box but we had to pay the Microsoft tax. Every piece of hardware we sold had-- we had to pay 60 bucks to Microsoft because the manufacturers had had this anti-competitive-- they later were found to be guilty of monopoly practices in this manner, they had to pay a tax. So that kind of-- you know, that thing-- and in order to be viable on an Intel, we had to deal with a bazillion different drivers out there. There was a huge space on Intel and so I put Objective-C into the kernel so that the kernel team could just subclass a new Ethernet driver. Just, you know, with inheritance and just tweak a little bit and a new Ethernet driver, <snaps finger> done. You know, and so that's how a team of 10 kernel engineers could put out a NeXTSTEP that ran on Intel, on many Intels, okay… that kinda thing. They later shifted that Objective-C driver interface over to C++ when they got to Apple. I still-- the kernel team still won't tell me who exactly did it. I think I know who did it but-- because Objective-C was unknown to anybody at Apple and they just feared it and they wanted C++. They made a C++ interface but they changed something in the compiler every release and so now they're using an outdated compiler because it's the only compiler that'll generate-- anyway. So I put Objective-C in the kernel and it actually helped them get into-- helped NeXT get into and stay in the Intel hardware business. My Distributed Objects and Foundation work got ported over to Solaris as something called Portable Distributed Objects. So it’s the Foundation layer, including the ability to go back and forth using, you know, object messaging, that got us onto Wall Street. Steve Jobs got up and demoed it with lasers. So, you know, effective application programming, Distributed Objects onto Wall Street, we eventually put-- ported-- I had it running on five different systems at one point.. Alpha, Hewlett Packard with reverse stack, Alpha 64-bit, Hewlett Packard reverse stack, Motorola, Intel and SPARC. I had a five machine-- actually I didn't have an Intel machine. I had four machines on my-- with real code that ran on it. It was packed fat and it would run anywhere. So that Foundation was a write once, run everywhere, if you recompile it, thing and that’s later what Java tried to do. Sun ended up buying at some point, rights, source rights to NeXTSTEP and that's what became OpenStep. So they had engineers poring over, trying to build a version of it, but they had access to all of our goodies. We had-- actually we reversed course in, uh, at one point we added yet another language extension around protocols, called categories on protocols. Bertrand came up with that, he cooked it up and we added it and we were retooling all of AppKit for it. I went off on vacation, I came back and Sun had bought our source code to the earlier version so we abandoned it.

Hsu: Oh really?

Garst: Yeah.

Hsu: So is that-- I mean, that's basically this--

Garst: Categories on protocols have been reintroduced in--

Hsu: In Swift?

Garst: In Swift recently, but they'd already been done at NeXT in 1995. It was on our to-do list which we never had enough staff to ever do.

Hsu: So I mean, you've been describing this whole sort of saga of NeXT, kind of…

Garst: NeXT was a failure!

Hsu: It was great technology but failing in the marketplace… I mean, what did that feel like and what was that.. how did?

Garst: Yeah, well, every 18 months the company would completely change direction and you know, it's Steve Jobs, you know, but he was dead on as a has been at that point. You know, NeXT is a failure, you know, it's great ideas, doesn't sell. Everybody loves it, nobody would buy it. But we had problems. I mean, at the first point he had a great little mission statement, which I have yet to find, but it talked about-- about two paragraphs-- he was trying to build a computer for the rest of us. It was really a Mac, it was a second Mac in his head. I don't know why he decided to build it as a ten-thousand dollar titanium cube. I'm not sure how many of my neighbors were going to be able to buy it. At that point five thousand dollars could get you a 286 PC. Ten grand wasn't too bad from that price perspective, but it was too bad to be successful.

Hsu: Right.

Garst: So it felt bad.

Hsu: Yeah.

Garst: I mean, I worked about eight years at Bell Labs without ever shipping a line of code that made it out of the system. And I worked on the UNIX, at the UNIX group and supervised people who did. And then I went to NeXT, and we shipped code that hardly anybody used. But it felt good. And so we just kept plucking along. So after Foundation, Foundation was a key point, and so we started getting more and

more successful after Foundation to some degree. But we would port to a new hardware platform, or then we ported to Intel, and then we got rid of the hardware and became just a software, an OpenStep company. But all of those, I mean, the Distributed Objects aspect, the memory management behind it was later emulated in Sun's and Java's RMI. But I got patents on that. I mean we knew how to do that. And at some point, Java came out in '94. And I looked at it and I go, "This is like a copy of Objective-C in many ways." You know, and it was! Because they had had Objective-C, and they'd looked at it, and Bill Joy had talked with us in the phase three about device drivers written in p-code, so you could prove-- you'd take the loops out and you could have provably correct device drivers. So I knew his interest in byte-coded engines from a long time ago. So I just saw all that stuff come together with Java, so I knew exactly what Java was. And it followed the same object model that Objective C-had, which is interfaces. Which Gosling onstage saying, yeah, they copied it. Because when we got out of the hardware business, our kernel guys ditched. Our kernel guys went and wor-- and got hired at First Person, which is the group that turned-- that made Java happen. I got interviewed for them. Or we talked to them, but they didn't have stock options. So I wasn't interested. But my kernel colleagues went there and took some of the same ideas. For example, the Java byte code marker that the JAR file magic number, the first four bytes of a JAR file are 0xCAFEBABE. Which if you look at it is a little juvenile. But turns out it had already been used. At NeXT, that same magic number was used to designate what we called FAT files. So you have an executable that's got the code for a 68K and an Intel and everything else, I mean, common data were necessary. That magic number at the top was that same 0xCAFEBABE. They took the same magic number and reused it, which is a cardinal sin. But that's what cowboys do.

Hsu: <laughs>

Garst: I did a cardinal sin, when I introduced garbage collection at NeXT.

Hsu: At NeXT?

Garst: I'm sorry, at Apple. In order to make it fast, I had to prove to Scott Forstall that it wouldn't slow down Safari, and so the garbage collector intercept point I used-- I broke a cardinal sin-- a cardinal rule. I hand-coded, or I had the compiler issue a hand-coded instruction to branch to a routine at an absolute address.

Hsu: Oh, wow.

Garst: So how many people get to deploy absolute addressed entry points in application code? I did. The instruction was branch and link absolute BLA, like my name, Blaine, almost. And it could only branch to the top part of core, or the bottom part of core. The bottom part of core was zero. Address zero was taken. So we branched to the high-end address, so there was a page or two of possible branch points for that instruction, so we took over the highest page of memory. And on the launch of any process, we do a

little quick check, are we running garbage collected or not? And if we were, we'd swap out the entry points up there that would go off and do what we'd call read barriers or write barriers for garbage collection. And if you weren't running garbage collection, it would simply return. So it was a branch and link and a return immediate, which was imperceptible in the Safari measurement, so we got to do garbage collection.

Hsu: Oh! Interesting. You mentioned the time when NeXT got rid of the hardware and moved to Intel. I mean, how difficult a decision was that on the company and on Steve?

Garst: Well, Motorola was not-- look, we did it twice. We did it at NeXT and we did it at Apple. Right? Same story. It was, Intel was just winning. You know, Motorola wasn't going anywhere the 68K product line, and wouldn't. And we couldn't talk ‘em into doing anything past the 68040. So Intel was the only other reasonable-- you know, and we tried all these other companies and different versions of software. And--

Hsu: But there was a RISC project at NeXT, though, right?

Garst: Yes. <laughter> We were using-- Intel had a RISC processor and we used it on that NeXTdimension board. It turns out, the IA-860. But there was a RISC OS. There was some talk, and there was a lot of talk about a NeXT RISC machine. And we never pulled it off.

Hsu: Hm. Just because it was decided that the company would be all software from that point?

Garst: Well, you know, I mean, you got a bunch of engineers looking at technology happen, and they have lots of ideas on what can happen. But yeah, I mean, an operating system is a huge endeavor. And we couldn't-- we had the RISC machine-- I can't tell you how far it got. I don't really-- I can't tell you with enough certainty to-- it went a fair ways. But it was far from complete when it was abandoned. My group, that I was doing Mach 3, right? I was trying to get Mach 3 up and running. To have a multi-kernel so I could have a-- go back and reintroduce my fast IPC and make it better and stuff like that. And there was just too much stuff going on. You know, and you didn't-- I mean we're buried, most software engineers are buried in bugs. And we had bugs. And we had projects, so the features, time, quality. You know, we did what we could. But we could cook up-- we had one tradition at NeXT that I think ought to be talked about is National Small Programming Week. So when we'd come up on a new release, Steve would designate, or software would designate a week of time where everybody got to try the new features. Just drop everything and try it out. It was sort of eat our own dog food. What can you do with this technology? And some of those apps actually ended up getting shipped, so like, at the moment I can't recall one. But they did. So we'd have a week of fun. You know, trying out the new stuff, and so everybody had to know how to program. So I went off and learned how to program applications. I was a kernel engineer. So I programmed a Go board little thing. And you know, so the-- at NeXT I was the senior guy and so I often

got involved in interviews, so I'd be brought in to interview various people. And the-- I introduced a-- we'd all gather. We'd have six or eight people interview somebody, two at a time sometimes. Three at a time sometimes, six to nine, nine/twelve people sometimes, on every candidate. And we really wanted to have consensus on our hiring decisions. And so we had, we established some rules, and I helped tweak-- and we talk about them every now and then. And so I'd help tweak some of those rules. Such that we always had-- you could have a veto. If there was something just really-- if you decided you would not be able to work with this person, that was enough, that was a veto. Otherwise, you know, we really wanted independent opinions. And so I said, "Okay, guys at the beginning of the recap, what we need to do is everybody give us a vote-- you know, do we like the candidate? Or do we not like the candidate? And so I introduced the Roman live or die, on the hiring decision protocol, as part of hiring. And then, of course, we ended up with midpoint-- not sure and stuff. Pretty soon. So then it became a clock. But the idea that everybody'd give their initial opinion first, and then we'd talk about it, and come to a consensus, meant that we would hire people who could communicate, who had good ideas, who had passion. And I was known as asking-- as giving hard interviews. There are people still at Apple who remember my interviews they had with me. <laughter> So one of them stopped me in the hall, stopped me in the cafeteria one day, and he says, "I remember that interview I had with you. I didn't think I was going to get the job." Like, "I remember that interview, too." <laughter> He goes, "You know, I mean, I was squirming. I didn't know what to do." And what had happened is, we're interviewing this guy, and he was fresh out of college and he had, you know, it was a great college, and he had done, you know, he had all his courses there, you know, and how do you get a program? You know, he was a sharp guy, clearly, you know, we wouldn't be interviewing him [if he hadn't been]. And so I go, "But why did you take this-- you took this class in artificial intelligence?" So I just decided to take a hard line on it just to see what would happen, because I was that kind of guy. So I go, "What'd you take that class for?" And it was like deer in the headlights, "Uhm.. now, I don't know." I go-- I pounced-- I go, "Well, it's a wasted opportunity. If you don't know what you were gonna do with that, you could have taken another class and learned something better. You know, and did you--," something like that, you know? "So what did you learn? Has anything happened in the past 15 years since I was in school? You know, have they gotten past," you know, "What'd you learn in the class?" "Oh, this and that." I go, "Okay, anything--," you know, and he was just struggling, you know, and he didn't tell me-- he didn't say, I mean, he could have said something-- I was curious, you know, and that would have worked. But he didn't have an answer for me. So you know, nah, it's all right, he's just out of college, but you know, so I cut him a break. And he asked back a few questions and realized I wasn't as up-to-date on it as I might have presented myself to be, and we moved on and we talked about other stuff. And so, you know, I recommended he got hired, and so he got hired, and so he was telling me about this in the cafeteria.

Hsu: Later on at Apple?

Garst: At Apple. And I walked off. You know, and it was like, "Oh." You know, what he didn't discover is that I had a long and abiding interest in artificial intelligence. So we didn't find common ground there, which I thought was a missed opportunity, but you're interviewing somebody who, you know, they don't need to know about what I'm interested-- I wanted to know what he was interested in, and what the passion was. And so I don't remember the timing of that talk in the cafeteria.

Hsu: In the Apple cafeteria?

Garst: In the Apple cafeteria, but it was Scott Forstall, who had led the iPhone and iPad development efforts, and everything. And I believe it was pretty close to the time they introduced Siri.

Hsu: Oh! Interesting! <laughs> Wow! That's a story! <laughs>

Garst: And I'll, yeah.

Hsu: So you were known as a hard interviewer at NeXT.

Garst: Yeah. Yeah. I don't—as some of my-- people still at Apple would describe me. They say I don't tolerate fools gladly. I'm a straight up guy. I talk with particle physicists. I mean, I got-- I mean, I'll talk with anybody. And I do. I've met the President, President Obama twice. I gave him three words of advice last time I met him. It was the only thing I said to him, and he followed them. <laughter> So, the last time I met the President, I gave him three words of advice, and he followed them. You know, so I'm that kind of guy. I met the President of my college, watched a basketball game with him in his box suite at the basketball stadium. You know? Talked technology and lots of stuff with him, and dark chocolate as well. Which we'll get to.

Hsu: Okay.

Garst: Yeah, I was at Batavia on the C-Standards Committee meeting talking to some of the parties. I talked to this physicist, he says, "I, you know, string theory, yeah, I don't believe in string theory anymore. I used to. I studied hard, and I wrote this paper. I came, up-- I pulled these numbers out of my butt, you know, and could predict results from the standard model, and I published this paper. I thought it was kind of a joke, and it was phenomenal! It got cited 200 or 300 times, and I had like two or three years worth of run off of that paper."

But it was a joke from his perspective, because he just pulled the numbers out of nowhere. He could push string theory to do anything. And so he and I, you know, I have interesting conversations with people.

Hsu: I want to go back to sort of the NeXT business story. So you mentioned Wall Street at one point. What was it like for NeXT to go into that business? I mean, you had mentioned that the original mission statement had been to create another Macintosh. And that didn't pan out. I mean, was it kind of antithetical to the mission to now be selling to Wall Street? I mean, what--

Garst: Well, yeah. But I had had some interactions with Dr. Richard Crandall.

Hsu: Okay. He was the Senior Scientist at NeXT?

Garst: Yeah, and he's big on Merseinne Prime numbers and still leads research in that effort. Largest Merseinne Prime numbers ever. And they had a particular application, an elliptical curve cryptography that he exploited. And so at some point we had bundled encryption into the system. You could just establish, so use it for public key, so you could establish an elliptical curve public key, and encrypt a message and send it to somebody, and you know, that you never knew before, but you knew their key and they could get it. And so the NSA came and visited us at some point, and I was in on that conversation, because I was interested in this elliptical encryption, and so I talked with him, and I got his source code and converted it-- it was in C, but it looked like Fortran, and then it got--

Hsu: Crandall's code?

Garst: Yeah, Crandall's code, and I think coded it up in-- I'm not sure I even went to Objective-C. Eventually I transcoded it to Java at some point. But I picked up and carried his elliptical curve stuff along and I had a hard-- I said, well, his first encryption thing, it was just after you establish the session key, it just dropped down to DES. Not even Triple DES. And DES was not very good. So I did a little bit of reading, I said, "Can't elliptical curves directly encrypt things? And it took a long time for him to understand the question, because he's very much a mathematics kind of guy, although he's actually a physicist, but he's more math than physics. But eventually I convinced him that, you know, you could use elliptical curves to actually do the encryption, too. And how come his stuff didn't do it? So he dropped by one day and left this book, Koblitz, elliptical curve, the guy who invented elliptical curve, wanted cryptosystems, he left me the book. So I'm digging through the book, and I got his code. So I'm trying to code up direct encryption myself, just because I wanted to. He says, "It'd be way too slow." And it didn't work. So he came by a few months later, I go, "You know, that didn't work!" He goes, "Really?! Okay." So we started talking about it. He goes, "Well, how are you doing the--," oh, I can't remember what they call it now-- the embedding, "How are you doing the embedding?" And I go, "Uh… what's the embedding?" He goes, "Well, didn't you read Koblitz? I go, "Yeah." You know, he pulls it out, and he goes, you know, "These pages." And I go, "Oh. I skipped those pages." <laughter> And he goes, "Argh!" And I go, "Well, what do you need to do that for? You know?!" And so he started explaining why embedding was necessary. And he's sitting up there on the whiteboard, you know, professor, you know, and he's drawing out this stuff, and suddenly he paused. And it's like one of those cartoon panels where the lightbulb goes up, and he turns and he looks at me and he goes, "Oh. Oh. Oh." And he discovered through thinking about it, just like that guy's program I debugged without knowing the language. I didn't really know the language of elliptical curve math. But he did. And I asked the right questions, and he figured something out. And he figured out that you don't need embedding. Or you need one bit of embedding. So the way he wrote up his elliptical curves was, if you negated-- so but the problem area is-- I'm probably getting way too deep into this, but the idea of elliptical curve is that you have to-- you have a curve, except it's not

really an elliptical curve, it's a three-- multi-dimensional. But to encrypt something, you take a point on the curve and do some operations on it, and you come up with a new point that's very unpredictable. But it has to be on the curve. It has to be a point that is on the curve. And so you can have a field that's only partially covered by the elliptical curve. So if you pick a point that's not on the curve, you know, you can do math to it, but it's meaningless. So you have to pick points that are on the curve, was the problem, the embedding problem. And so he figured out that if it's not on this curve, it has to be on the negative of that curve. And so it's always-- it's embedded on one curve or the other. And so we got a patent out of that. And I added some value to that patent. So I have a patent in elliptical encryption. But it turns out that some elliptical fields have, I believe, complete coverage, so that there is no embedding problem for some fields. But anyway, so did a little bit of elliptical encryption work along the way. <laughs>

Hsu: Okay. Did you have any ethical qualms about doing work for the NSA or for Wall Street banks?

Garst: So the reason I did not work for the defense company that offered me a hardware job and a software job was I didn't really want to build bombs, or defensive stuff. And the NSA was using NeXT equipment to do, you know, monitoring. They had lots of-- they were using Distributed Objects, you know, they had lots of machines. They were trying to do graphics visualization of all kinds of stuff, and you know, they were paying some of our bills. And so Wall Street, Sina Tamaddon got us onto Chicago options trading market. These guys, they would spend 20 million dollars on new technology to see if they could get an edge up on their competitors. And so if they went and bought a hundred NeXT machines and spent six million, or eight to ten million dollars writing programs for it, and it won, it was a win. And if it wasn't, it was a loss, they'd go onto something else. They were in a different league. And they were happy with NeXT, they liked it. And so when they got Distributed Objects, which let them run their database operations on a Solaris machine, or an HP machine or an IBM machine to do their frontends on an NeXT terminal, that was a big win for them. And so that's what they did. So when Java came along, the languages were so similar that, and we had our WebObjects technology, which sat upon our EOF, Enterprise Object Framework database technology. I came up with this crazy idea that we could have a Java bridge. That we could expose Java APIs that were really built out of Objective-C, using Java's JNI hooks. And we could do it sort of mechanically. So we wrote this description file that would take a description of any of our Objective-C frameworks, and put in some naming changes, and so the JOBS file-- J-O-B-S-- Java to Objective-C, JOBS, <laughter> would take Objective-C, would build a Java spec. It would build all the Java necessary to glue onto that framework and expose it in Java. And so in 1996, we had, using WebObjects, we had a Java-- we had a web server using Java based on top of our Objective-C, using this Java bridge technology. And it was that technology, apparently, that we had a Java story, and then we were going to go public with that. That was what the company eventually found was going to make enough money on to go public with.

Hsu: NeXT?

Garst: Was WebObjects on top of my Java bridge technology, which was on top of the Foundation, which was on top of-- I mean, I look back at these techno-- technologies, these things that I put together for the company because it was fun! And it was the right thing to do. But I-- some of these ideas carried NeXT forward. And at Apple even, I added a few ideas that carried them forward. So I like ideas. So I still have ideas, and so I have even bigger things to do with them, if we ever get to it.

Hsu: Okay. Maybe if we could step back and talk about your wife that you mentioned earlier. How did you meet?

Garst: So I met my wife, Carol, at an offsite meeting. Bell Labs, who had lost this class action lawsuit against the operators. The operators sued and said, "We're being discriminated against. And so they won. And so AT&T was obligated to provide sensitivity training around affirmative action, and to all of its employees. And so they were very rigorous about that. So once or twice a year, we'd have an opportunity to take a week or two and go to different courses. So I took courses in time management, I took courses in all different kinds of stuff. But every now and then we'd have sort of an HR-directed, a affirmative action kind of meeting, and we had several of those, and so this one, in particular was about-- had a bunch of newly minted supervisors, men exclusively, and women, all the secretarial support staff, admin-- it was so obvious, right? You just put these people together in the same room. So I had already, you know, I'd taken a class in economics on this, right, years before, so I already knew the situation. But anyway, so I go into this offsite for affirmative action, more or less. And I was a young guy, a supervisor, 26, and--

Hsu: What year was that?

Garst: Oh, I don't know, '82? I guess? Yeah. And this woman walks in late, you know. And she sits all the way across. And the first exercise was, "Pick somebody you don't know, and go ask each other some questions." And me, not being the bar scene kind of guy, I had a fit of just absolute boldness, and I looked across the room, and she was looking at me, too. So we went and did that.

That was-- I had seven years with my wife. She had a heart condition, mitral valve prolapse, fairly benign. We knew about it. Generally, you go to your-- you take an antibiotic before you see your dentist. And I don't know, a couple other things. But we'd had a kid. And she was pregnant with our second child. He was about one-year-old, we had a big Father's Day-- our first Father's Day together, you know, Mother's Day together. Her birthday, my birthday. Had my family come in. You know, it was a big deal. And then a month later, her heart failed on her. Pretty much right next to me. And did everything we could, and there was nothing that could have been done, and she was gone. And so my life went from here-- it went that a way. And it was hard. Hardest thing was that our son was 13-months-old. And he didn't know anything about what was going on. And I realized that I had to be genuinely happy and joyful for him in order to give him what he needed. And he would look up to me and smile, and that smile was not meant for me. It was meant for us. It was a joint project. I'd get up with her and help, you know, moral support. She nursed him in the night, and I'd change his diaper. I was a very involved dad. And so I was in that sense, trained,

I guess, or prepped for her loss. But she was a very happy, joyful, caring person, and she left-- she made an impression. We were fun together. We were very different. She wasn't very technical. She was an English major. But you know, so I went to my high school, I mean, I only chose to be a computer scientist because it was fun. So-- I write well. So I could deal with that. We had a lot of fun. And we did as much as we could together. And so when I got up, she passed away at 11 o'clock at night. And at 11 in the morning, I'm buying a-- I'd called about 17 different relatives through the night. And I'm sitting at the cemetery buying a funeral plot, you know, buying funeral plots with her parents there, you know? How many plots do you buy? "Well, are you gonna get remarried?" I mean, goddam, I mean, geez, you know, worst day of my life! Worst day of my life. And two days later, on a Sunday, I'm up there and I smiled. And I said, "I had the best time! She gave the best of her life to me." <pause> And my regret was not-- I had no regrets about the time we had together. My regret was that my son would never know her. So a bunch of people from NeXT came. Couple years later, her mother passed away. She was the local support system. I mean, she and her-- my father-in-law, mother-in-law lived in Sacramento, they were helping me out, and then her mother dies of the same thing! And it looks out like her grandmother had died of the same thing, too, in looking back. So it looks like a hereditary thing. So I kind of had to raise my son on my own for the most part without any local support. And my best friend, then, he was-- I had recruited-- he had been at Bell Labs, out on Menlo Park, one of my colleagues. Or he had went to Apple. I went to NeXT, he went to Apple. And he wasn't-- he was in Developer Tools even. And he didn't like it too much there, and I recruited him to NeXT. So he'd come to NeXT, and in fact he had become my boss, as it turns out, because I was doing a lot of Tech Lead stuff. And you know, you can't do Tech Lead if you're a Manager. You either manage or you code. I can't do both. So I was back to coding. And he came down with leukemia, and two years later, he passed away from that. So I had a succession of-- and it doesn't stop there-- there's more and more. So my personal life has been one of stoicism, of making the most of every day, and living life without regret. So I live life with complete enthusiasm. I own three Tesla's. <laughter> Including the original, the two-seater roadster, which is what I drove here today. So people came up to me-- one guy came up to me in the parking lot at NeXT a year or two after, and he says-- I didn't really know him-- and he said, "I admire you." He says, "I couldn't have done what you did. And I just think it's amazing!" <pause> I don't know what I did to be able to smile to my son. I don't know where I put that pain. It's probably still in there, because I did what I had to do, with a smile, with enthusiasm, with genuine enthusiasm. And it was after she died that I went on to do the Foundation work and a bunch of other stuff. So I, you know, I balanced raising my son with contributions, but you know, I sacrificed some things, too. There's things, I never decided to really rise up in the management chain, for example. You know? I didn't have 80 hours a week to dedicate to work when I had a son to raise. So that affected me. So yeah. You have to be prepared for left turns. And that was a big one.

Hsu: Did you ever remarry?

Garst: I dated furiously, honestly, because the idea of raising my son alone, just of him not having a mother just was terrible to me. So I didn't ever-- you can't heal from that kind of a loss. You can only recover. And so I dated. I got engaged a couple of times. And that didn't work out. So I've been father to some of my girlfriends' kids for various times. And when they break up, when they go away, that's another kind of a loss. So I'm at the point now where I give back a lot. I've given back to my high school. I've

helped-- I give away laptops. I got my blood drawn at Apple. And the person drawing my blood was just really personable, just really, you know, caring, I could tell. So I'm always talking with people, and chatting and stuff. I'm an introvert, Myers-Briggs, INTF-- INTP.

Hsu: Oh, I think that's what I am.

Garst: But I'm a trained extrovert. So I asked her out. So you heard've Apple? You know, "Do you have an Apple?" She goes, "No, no, I don't." "Oh, yeah, so you have a PC?" She goes, "Well, actually, no." I go, "What? You don't have a computer?" She goes, "No, no, no." And I thought about the three or four old computers I had just sitting in my house, I go, "Do you want one?" She goes, "You-- really?!" And I go, "Yeah. Yes." Well, give me your email and stuff. So I took down her email and stuff. Well, I had a girlfriend at the time. And I didn't quite know how to explain this to her. <laughter> So it took me a few months to actually bring it up again. Because the laptop I was going to give my girlfriend, she didn't have any interest in. And so that was the laptop I was going to end up and so-- it went on, it took me almost maybe eight months, before I actually decided, "Well, dammit! I'm not going to wait for her to dec-- my girlfriend to decide on this thing. It's wasted." So I called her up again, and I drove up, and drove her down to Palo Alto to the University Lab, and looked over-- talked it over, and I bought her a laptop. So she used that laptop. She had two kids. And so the kids got to use it a bit. Her husband had a PC, which they'd been using, but it was pretty well dedicated. So that was the spare computer for their household. And she used it to get a better job at Kaiser doing different kind of nursing. And we're still friends. And it was just a random act of kindness. <pause> Tragedies happen all over the place. One of my cousins had a son, who was crushed to death while repairing a suspension on an ambulance. He was about 32. His wife was about 30. Had a four-year-old daughter, and she was pregnant.

Hsu: Oh, my god.

Garst: When I heard that, you know, I'm at the point where I can reach out like Steve did to me, and say, "Do you need anything?" So I do that. So what kind of man was Steve? He was inspirational. So I've gone back to my high school. I hope that for a lecture, or a week of teaching, maybe I can do for some of those kids what Ella Leppert did for me. Give them deep ideas on how to rethink the world. 'Cause unless they get engaged in solving the world's problems, you know, some of the smartest kids on the planet, then who will? So I'm trying to teach those messages again and again as I see them. So I've healed myself since my last relationship, and so who knows?

Hsu: The world is open?

Garst: Yeah, yeah! I have no idea-- well, I do, but you know, I'm open to the world. I'm making friends all over the place. Every country I visit, I make some new great friends and it's a lot of fun. And I have a Save the World project that I work on to keep me more engaged.

Hsu: Right. How old is your son now? How's he doing?

Garst: He's 25.

Hsu: Oh, wow.

Garst: He drives one of my Teslas. <laughter> He ended up in Computer Science, and designed a 3D printer for his senior project.

Hsu: Wow.

Garst: And he and I bought a-- I remodeled part of my house, and put 108-year-old pool table and it's the centerpiece of my art room. It has a lifetime-- free lifetime warranty for leveling, 108-years-old.

Hsu: Wow.

Garst: I can get it leveled any time I want.

Hsu: Wow.

Garst: He and I designed and built a LED light strip pool table light out of tap plastic, 4 x 8 sheets and some aluminum struts, and some Arduinos and stuff. So you do a little touch panel here, and random lights show up. Or we can do the color mix between the two different hue spectrums that we do on that. So we have fun. We have fun.

Hsu: <laughs>

END OF THE INTERVIEW

