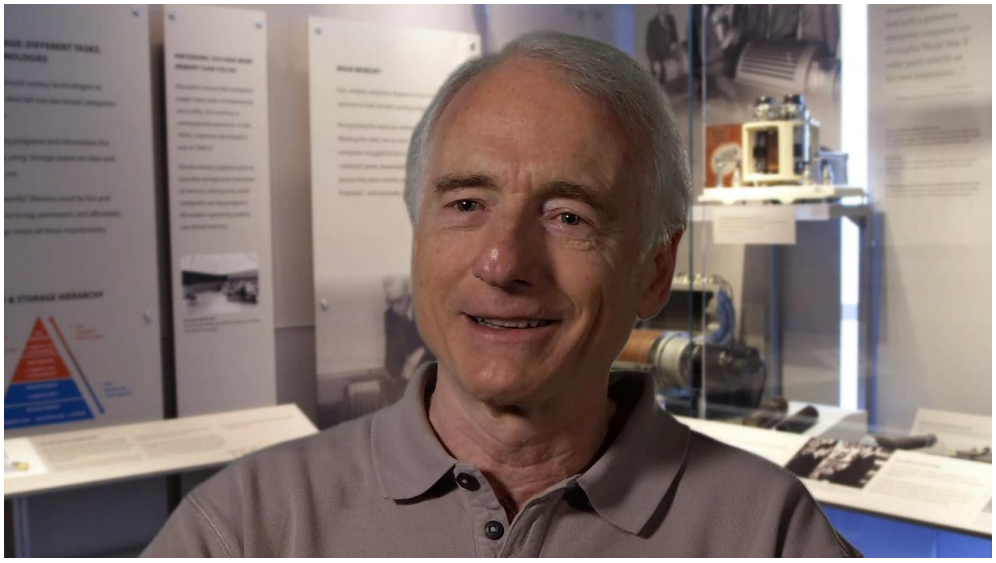
Oral History of Lawrence G. “Larry” Tesler

Interviewed by:
Al Kossow

Recorded:
February 12, 2013
April 22, 2013
Mountain View, California

CHM Reference number: X6762.2013



Larry Tesler

Kossow: Good morning. This is Tuesday, February 12th, 2013, and I'm Al Kossow, the software curator at the Computer History Museum and we're talking to Larry Tesler who I've known for a very long time.

Tesler: Yes, long. <*laughs*>

Kossow: You were from New York originally?

Tesler: I grew up in the Bronx.

Kossow: Do you have any siblings or—

Tesler: Yes, an older and a younger, and the older lives in California; the younger lives in Washington State.

Kossow: You originally got interested in computers in high school?

Tesler: I went to the Bronx High School of Science and I've been interested in computers ever since the 1952 or '6, whatever, election— presidential election where they tried to have a computer forecast the outcome and I was just fascinated about—

Kossow: You watched that on television?

Tesler: Yes, and that fascinated me and I wanted somehow to get to use one of these computers so I wouldn't have to manually compute things. We didn't even have pocket calculators—slide rules and paper and—

Kossow: Were you good at math in school and—

Tesler: Yeah, until somewhere in the middle of college I was really good at math <*laughs*> and that was— and I even got my bachelor's degree with a major in math at Stanford.

Kossow: I guess you had some contact with computers when you were at the Bronx High School…?

Tesler: Yes. So what happened was that, as is my personality, if I ever hear somebody say something's impossible or extremely difficult, almost impossible, it's a challenge and I always try to do it. So I used to try proving Fermat's Theorem. I would just try to do anything that couldn't be done and learned a lot along the way and when I was younger hardly ever accomplished anything that was in that category but trying taught me a lot. But in this case there was common wisdom in the math community that there was no closed formula that generated primes and only primes. I thought that just can't be right. So I went off and came up with one that actually did. And the story goes a little differently; if you’ve never heard about Tesler's Formula it's because it had a flaw in a way. But it really did compute primes and only primes, in fact *all* primes. And I took it to my math professor and showed it to him and showed him the proof and he said, "This is pretty impressive. But you know, this isn't really what they would call a closed formula. It's called an algorithm." “An algorithm? What's that?" And the only algorithm I'd ever heard of was Newton's Algorithm for computing the slope of a line tangent to a curve and I didn't really know what it meant so he explained it to me, and he said, "But what it means is that you could program this on a computer and it would generate a lot of primes, all the primes." I said, "Well, how could I get access to a computer?" We don't have any here at the Bronx High School of Science in 1961 and— or 1960 actually at that point and so he said, "I don't know but I can get you a manual and try to figure out some way you could get access to a computer." So he gave me an IBM 650 machine language manual and I took it to the cafeteria and started reading it and while I was reading it a student stopped by and said, "Where'd you get that? Why are you reading that?" and it turned out to be Paul Schneck who later was in charge of the on-flight electronics and software for the space shuttle and at that point he was a year ahead of me I think in— at Science and he had been getting computer time at Columbia. So he arranged for me to get computer time at Columbia also, a half hour every other Saturday. That was, it turned out, just enough time to take my deck that I punched on the card punch, put it in, run it through the three passes of the compiler and find one syntax error or maybe one bug in my program and then two weeks later I could try again. So I could go into detail on all this, but that's what got me into computers and the moment I got to Stanford the following fall, the first day I was there I ran into some other math students in the dorm who said they had access to computers at Stanford 'cause they'd gone to high school in Palo Alto and they were in a special program. One of them was Charlie Brenner who is now a forensic computer person and analyzes DNA of people that die in mass disasters, things like that, to try to identify who's who, and the other was Doug

Hofstadter who probably doesn't need any introduction. We met the first week of school and we have been friends ever since. But one of the two of them got me over to the computer center and I met the— two of I think three computer science professors at that point, there wasn't even a department yet, and they said, "You could have all the time you want on the 650. Nobody else uses it." <*laughs*> So I did that for a few days and then started wondering why nobody else uses it and discovered a Burroughs 220 a couple of rooms away. It had tape drives and all sorts of amazing things and so I switched over to that, became a computer operator so I could get programming and free computer time when nobody had other jobs to run and kind of embedded myself in the computer science milieu at Stanford pretty deeply.

Kossow: How did you end up on the West Coast?

Tesler: I wanted to get away from home. I didn't have a bad relationship with my parents in any way I mean but it wasn't really a good one. They were kind of constraining and over-controlling and I was just too free-spirited for that and so I just wanted to get a distance away and also I had been mugged a couple of times on the streets of New York and I just thought I wanted to go to a place that was a little safer. So my uncle, my father's brother, lived in L.A. and during a visit my parents or I asked him, "Are there any good colleges in California?" and he said, "Well, I heard of a place called Stanford that people say is really good." We'd never heard of Stanford and so I looked into it and ordered a catalog and looked at the catalog cover, and it was beautiful young people in shorts, sitting on grass <*laughs*>—and I went, "Whoa, this beats Columbia or City College or anything else." So I applied there and Columbia and Swarthmore, got in all of them, but I just wanted to go far away so I insisted I was going to go to California.

Kossow: The card stunt was— you were a sophomore—

Tesler: Well, when I was a freshman that fall the— I had been to some football games and participated in the card stunts and at the beginning of the fall season. We had these handwritten instructions and for the last three games or something they were printed instructions.

And so I was curious about that and asked around at the computer center and one of my friends there actually, Larry Breed, said, "Oh, well, that's because they're using our software now" and so he introduced me to this Card Stunt software that he and Earl Boebert had done. Larry Breed later won a Grace Murray Hopper Award for his work on the first APL implementation. But in 1961, he was my mentor and for the next couple years he taught me good practices of programming. He had the best practices I still have ever seen, and so he taught me those. And I became a very good programmer. And did various projects at Stanford, at first for free, and then became a consultant and charged people for my time as well as being a part-time employee at Stanford working at the computer center.

Kossow: Were there very many other people outside offering programming consulting services?

Tesler: When I— I incorporated a company called Information Processing Corporation, <*laughs*> a very kind of over-the-top name for a little company with, at the beginning, just me and eventually four people or something— five— four other people and I think we were the fourth or fifth, maybe sixth data processing company is what we called it then— or what the phone company called it then in the Palo Alto phone book. There was a subsidiary of IBM, Service Bureau Corporation. There was a guy named Richardson who—

Kossow: Richardson Data?

Tesler: In Palo Alto?

Kossow: Uh huh.

Tesler: He actually— wow.

Kossow: We have some stuff in the collection from Richardson Data.

Tesler: Was— it's a small Palo Alto software company, maybe a dozen people when they got big—

Kossow: Uh huh.

Tesler: —and he just retired a few years ago. I happened to stumble upon him at Town and Country Village just as he was packing his stuff to retire and— whoa— what was the other guy's name? You might know it, <*laughs*> just totally blanked on it for a second there, but there was another guy who was a Stanford student also, a little ahead of me, and he later went on to develop AppleWorks and so we can check that name later; I'll get back to you on that. Rupert Lissner is what it was, Rupert Lissner, and he had a consulting company and his customers were all businesses and he did business software and so we shared an office on Ramona Street in Palo Alto for a couple years and the phone calls that came in to either of us if they were business software I would hand it to him and if it was education software, graphic software, consumer— whatever— scientific research software, statistical software he would hand it to me. So I got what I thought was the interesting jobs and he got what were the lucrative jobs <*laughs*> and so eventually we outgrew a shared office and split in to separate rooms but stayed in touch over the years and I ran into him again when he was doing this AppleWorks project for Apple.

Kossow: You would write programs for people who had their own computers? You didn't have a computer.

Tesler: Right. I mean you couldn't have your own computer at that point. I don't think anybody had a computer of their own. Companies owned computers, universities, government agencies, military of course, and I had first done that— taken over the Card Stunt program from Breed and Boebert and improved the usability of it and that was 1962 when I was a sophomore. And that's when I started doing usability testing with art students at that time who were drawing the frames of the animation and I wanted it so they would keypunch— or prepare the coding for the keypunch forms instead of me having to do it, so I kept making it easier and easier for them to use over the years. And that gave me a little bit maybe of a reputation and so the first couple assignments I got from departments within the university, people who wanted room scheduling for the psychiatry department at the medical center, statisticians that wanted analyses… Any time there was something involving not necessarily interaction, 'cause there wasn't a lot of interactive stuff then, but anything involving the writing of language statements or entry of data onto forms I would focus on making it really easy to enter that data or to learn the syntax of that language. And usually I did it in collaboration with the client and jointly designed it and then I implemented it and then tested it and they tested it, we improved it, so I kind of got into the usability side of computing pretty early. I'm not the only one who was doing that. Larry Breed, in fact, is probably the one who got me most excited about it but I'd already been focused very much in my mind on making computers easier because one day I wanted to have a computer; I knew that was going to be sometime in the future. At the beginning, Moore's Law wasn't there yet but after a couple years of being at Stanford, Moore's Law was declared and it became clear to me that I was not going to be that old by the time I could have my own computer and so— but also for that to happen there had to be— it had to be something everybody could buy or I wouldn't be able to afford one so I started focusing on anything I could do to do the projects that were handed to me but also do it in such a way that I would either learn more or push the industry more towards ease of use.

Kossow: At that time you started doing some contracting with SAIL around 1967 or so?

Tesler: Yeah. I think the SAIL contracting started maybe '66. I graduated in '65 and at that point I'd been running my business for about two years, the incorporated business, and SAIL was one of my clients, the Stanford Artificial Intelligence Lab, and specifically Ken Colby who was a little different from most of the people there—but there were a lot of people there who were different from most of the people there, they were working in computer security, they were working in computer music—but Ken was a little closer to the artificial intelligence mission. He was working on simulation of cognitive processes, being a psychiatrist. He was one of the founders of cognitive psychology and so among his network of people were Don Norman who came and visited us from UC San Diego a few times. After Terry Winograd wrote his thesis he came out to visit the Stanford AI lab and we met him. At that point it was probably the most practical and interesting application of— in AI and so we were delighted to meet him, and he had a lot of interest in the cognitive psychology aspect of AI. So we did some interesting things. I was focused more on natural language understanding than cognitive modeling. I did a little bit of designing a language for cognitive modeling and— but in the natural language [area], we interviewed a guy named Roger Schank and hired him and that was his first job after his Ph.D. and he and I collaborated and both of our names were on a few papers in the early days there in the '60s.

Kossow: Did you have much contact with the people at SRI and Engelbart's lab or—

Tesler: Oops. SRI was one of my early clients. When I was an operator on the Burroughs 220 there were these civil defense simulations and what they were was if a bomb was dropped on a city somewhere given the prevailing winds and all that sort of stuff, a nuclear weapon, where would the cloud— the mushroom cloud--go and what would get contaminated. That was a major concern at that time, the middle of the Cold War, and so SRI had a contract to do these simulations and they sometimes couldn't get time on their own 220 'cause it was too busy so they'd send it over to Stanford and we'd run them overnight and it was pretty much just me and the SRI guy there all night sometimes, didn't need a security clearance or anything, but it was not as public as most of the things we were doing. And they then asked me if I could program and <*laughs*> they had some changes they wanted to make to the software and a programmer wasn't available or something so I started doing that and did some other work for a couple of people at SRI, but I actually didn't know anything about Engelbart until after the big, famous demo that he did in what, '68?

Kossow: Uh huh.

Tesler: One of my clients, who was literally a consulting client— I mean he would just come ask me my opinion about all sorts of things. He didn't have me do any code for him at all, just— 'cause he was a consultant in a more general business context and he needed somebody who was technically expert to— as a sounding board to discuss these future trends and other things. And he came to me one day and said, "You have to come to SRI with me. I need to show you something" and he had gone to the big demo, he had no idea what he was getting into, and after he saw it he dragged me over to SRI and got me a personal demo. So that was I guess just a few weeks after the big demo and I had an instant opinion about it, which was the power of it was amazing and I was seeing the future, there was no doubt about it, but the user interface was way more complicated than I could imagine teaching to somebody and it involved a mouse with three buttons on one hand, five-key chorded keyset on the other hand and many, many, many modes. And in those days all interactive software, and there wasn't much of it, had modes and it was considered a necessary evil; you couldn't not have a mode. I mean how could you do that? Some people disagreed with that. I was one, Dan Swinehart, who was in the next office from me at SAIL also agreed, but most people just thought well, that's just the price you have to pay to get all this power on a computer. But SRI was taking it to an extreme. Every keystroke started a new mode and every click of the mouse.

Kossow: Explain what a mode is.

Tesler: A mode is a state of the system that influences the interpretation of other things that you do. In particular, in those days the main issue was if you type a key on the keyboard like the 'D' key does it type the letter 'D', does it type the capital letter 'D', which is— which it would if you hit the shift key or the shift lock key or the caps lock key which you might forget you hit and get the wrong type of 'D', or is it an

abbreviation for the delete command. And a little later we were at— when I was at PARC, Charles Simonyi and Butler Lampson built this editor— and Tom Malloy and a few other people— called Bravo and it was fairly low on modes at the beginning and even lower over the years, they would keep reducing the modes, but if you typed the word "edit," which would be a common thing we would type because we were building editing systems, 'E' was shorthand for "entire". And so if you were in command mode and not typing mode, 'E' would select the entire document. And then when you typed the 'D' in edit [mode] it would delete the whole document. Then when you typed 'I' it would insert— go into insert mode, when you typed 'T' it would insert the 'T', and then you'd look at the screen and all you'd have is a 'T', and there was an "undo" but it only went back one step so you lost your whole document unless you’d saved it. So that was an extreme example but in Engelbart's system, which was called NLS at that time, mode switching was just constant. It was actually the power of the system; they thought of it as the power of the system. You could have a huge amount of power because you weren't limited by the number of letters in the alphabet. You could enter modes and reuse those same letters to mean different things but if you were looking at the screen it wasn't such a bad thing because you might be able to tell what mode you're in. The feedback wasn't so great so you couldn't usually easily tell what mode you're in, you have to look at the bottom of the screen, think a little bit about it, but it was possible. Certainly, if you started typing "edit" you stopped pretty quick when you saw your whole document go away. You probably— maybe wouldn't get around to typing that 'I' and they didn't have an undo command so it was important, But they just never really tested alternatives. They tested it— they brought in users and they tested it but they never presented to the user an alternative that didn't have modes and they just stuck to their belief that that would be way too limiting. And Engelbart would even say, and even today still says I think, that if it takes six months to learn— not that it was that bad but— if it takes six months to learn to use this as a power user it's worth it because it's going to augment your intelligence and give you all this power and it's less time than it takes to learn a language, that kind of thing. But I just had a totally different attitude about it, which was why have people spend six months to become a user; why don't we spend six months or six years even if that's what it takes to make it really easy so people can learn it in six hours, not six months. And in fact, the sort of things that we were doing at that time, today, could be learned in six hours if you get all the software that duplicates what Engelbart software could do, which was again very powerful and it's not even standard stuff you can get today.

Kossow: Did that influence any of the work that you did then at SAIL after you saw that? That was sort of—

Tesler: Yeah.

Kossow: —around the same time.

Tesler: It influenced— well, it gave me a view of where things were going in a dimension I hadn't thought of before and I just perceived things differently after that. We had a pretty primitive display at SAIL in

comparison—it was a Character Generator— I don't even think it had any graphics on the PDP-10 that we were using. The PDP-1 had graphics because it had *SpaceWar!* but—

Kossow: These were the vector displays made by Ford?

Tesler: That sounds familiar. They were the vector displays, yeah. They were called the data— point data— no, DataDisc. I can't remember the names of many of them. It could have been a subsidiary of Philco Ford or a competitor, don't know, but it— when I would think of things to do I often come to the point of saying, "Well, it really can't be done in this display or with this set of input devices," which was pretty much just a keyboard at that point with a lot of extra keys. "I'm going to have to wait until something like Engelbart’s system comes down from $100,000 a seat to a thousand dollars a seat and then I'll be able to do some of these things," and again with Moore's Law I thought this can't be too far away but— so I was just brewing ideas and waiting until the hardware could implement those ideas.

Kossow: This was around the same time you and Jim Warren were putting together the catalog for the Free University—

Tesler: Right.

Kossow: —in the cut and paste.

Tesler: I was a member of the Midpeninsula Free University. It started around '66 or something. I joined it I think '67, '68, and it had a course catalog with hundreds of courses and also a magazine, kind of. I think they called it a newsletter but it was more like a magazine format where people would write articles and then we would get somebody to type them up on a typewriter and then cut the— these galleys essentially into little pieces and then— with scissors and then take Elmer's glue or something and paste them up on pages and then send them to the printer, which was originally an outside printer but we actually bought a printing press and had a press operator so we could print our own catalogs, actually saved money over the long term. And one of my partners in paste-up was Jim Warren who was a math teacher at that time in high school and one day we were sitting there doing the paste-up and I stuck something down at the wrong angle or something and I went, "Ah. Someday you'll be able to do this on a computer and there won't be any mistakes. We can make corrections and it'll take a lot less time. We won't have to do all these little manual alignment things" and he said, "Huh?" Even though he was a math teacher, he didn't really know what computers were about, never really had seen one or if he had it was a business computer sorting checks or something. He had no concept that it could do what I was talking about so I said, "Oh, I got to show you *SpaceWar!*" so we went up to the AI lab and I showed him *SpaceWar!* and he gradually got connected with the AI lab, left his job teaching and entered the computer industry, started the West Coast Computer Faire and Dr. Dobb's Journal and so on, but— so that's how he got into the industry. But as I was explaining it to him, I started going "Yeah, really, this would be pretty

cool. We could edit the way we usually edit and then we could paste up on the screen and then send it to a printing press if there was a way to get it from a computer to a printing press. I don't know how to do that but line printers (which is what we used at the time) wouldn't be good enough. Again I saw it as something that the hardware would eventually enable and when it did I wanted to do this; this was a goal I set for myself.

Kossow: At that time you were teaching as well at the Free University?

Tesler: Oh, at the Free University, not at Stanford but at the Free U I taught a number of classes and sometimes it was— when it would be two o'clock in the morning and we were still pasting up the catalog we'd get silly and start writing nonsense course descriptions and if we still liked them by the time we were— at quitting time we went, "Oh, okay, let's put that one in. I'm willing to teach that one" and so Jim and I both did that and we had a few courses that were dreamed up while we were joking around about what course wasn't there yet. The first course I taught was a sort of serious one. It was 1968— fall 1968 so it was the summer and it was for the fall— I mean we produced it in the summer for the fall catalog— and it was called *How to End the IBM Monopoly* and it was a little blurb about how IBM dominates everything and it's bad for the— it was bad for the advance of computing and bad for everybody 'cause prices were too high for people using computers and we needed to end the monopoly; anybody who wants to talk about how we could do this come to this class. So that fall we had a class of about eight or ten people. We'd meet every week and we would talk about how hard it was to use IBM software, which was my main beef, how the price kept going up, they would— they'd get you hooked on a piece of software that was $10,000 a year and then the year after now it would be $15,000 a year or they raised the price of computer time at their service bureau, that kind of thing. They had a monopoly, they could do anything they wanted, so about halfway through the course— through the fall quarter, a couple of the people in the class admitted that they worked for IBM and they were sent to spy on us and they felt bad about it 'cause we weren't really doing anything malicious; we were really— had goals to try to improve things and we thought the only way to do that was to end this monopoly. And they said, "We're conflicted. We don't want to keep coming anymore so if you don't see us it's not because we don't like what you're doing. We can't obey our bosses" and so they stopped coming. Les Earnest of SAIL told me later that there were even more than the two that confessed; there were other IBM'ers there; it was probably mostly IBM'ers. So that was fall of '68 and I scheduled a second quarter of it 'cause it was starting to get kind of interesting. We were coming up with all these ideas and we had a list of accusations and we thought we should put together a whole list and we'll submit it to the Justice Department. The week of the first class of that quarter— on January 19th— the Justice Department filed suit against IBM, antitrust suit, and well, <*laughs*> I guess we're moot now so we canceled the class, said, "We'll leave it to bigger guns to take care of this."

Kossow: You went to work full time for SAIL for a while.

Tesler: Yes. They were one of my best clients and in 1968 something happened, which was we had a recession, and recessions hadn't really affected the computer industry up to then because the computer industry was very tiny and growing very fast but still very tiny and very few people were employed in that and it just wasn't an effect, but this time there were enough people in the field that it actually made a difference. And a consultant was one of the first things companies would cut when the recession came at that time. It may be different now but that's the way it was then so suddenly all my work dried up and I had mentioned that to Horace Enea at the AI lab who worked for Colby also and he said, "You know, I think we could talk about hiring you." So they created a spot for me and I joined the group and started working as I mentioned with Roger Schank a little later but from the very beginning with David Canfield Smith who later was one of the principal designers of the Xerox Star and Horace Enea. Horace became a venture capitalist internally to Apple at first and then outside Apple. But in those days— or in between those days and when Horace joined Apple, he worked for Citibank which was really revolutionizing banking with technology; he learned a lot there, and then when the Apple II came out he and some friends put together speech recognition software for the Apple II, brought it to Apple, gave a demo to Steve Jobs, Steve handed them another computer that they could use for development, and they went ahead and developed this amazing software for the time. And so I worked for three years— I guess two years at the AI lab and then one day I woke up and said, "It's going to be a long time before computers can actually do anything that I would consider intelligent. It's going to be many— past my whole life and if it's going to take 25, 50 years I'm too practical. I don't want to do that." I had this consulting business before and did practical things that people would use right away and that's what I came here to do. And this cognitive modeling is kind of right away, but as I was being drawn into natural language understanding and other aspects of AI were popping up around me in the AI lab, I went "All these things are way in the future. I— it's just not for my personality." So I decided that I would do something more practical so first I just dropped out of the field entirely 'cause I didn't think anything else was interesting but AI and I went to Oregon and I lived with a bunch of other people in a rural setting and learned about doing that, living in the country— I'd never done that; I grew up in the Bronx of course— and that was pretty cool but there was no income up there. I went to a bank and tried to get a job and they said, "We don't hire people who have programming background to write our software. We hire tellers and it's a career path for the tellers." I went "Okay, but— I won't get a job here" so I went back to the AI lab and I asked them if there was anything I could do there and first they mentioned, "Well, Alan Kay recommended you to join Xerox PARC." I said, "What's Xerox PARC?" and they said, "Oh, just a couple of weeks after you moved away from here Xerox opened a research center in Palo Alto and Alan Kay doesn't work there yet but he's planning on working there and he's been helping them hire people and he recommended you but nobody could find you." I went "Oh."

Kossow: When did Alan come out? Was it just after he got his PhD?

Tesler: He and I were both at the AI lab together maybe a year before I left so he probably came in '69 and I left in June '70, came back in maybe October '70 and sort of missed the opportunity to be one of the first people at PARC so I went over there and met the dozen people that worked there or something at that point and talked to them and said, "I would love to work here but I'm— I live in Oregon and I love it up there and so I need to be able to work remotely through this new ARPANET thing that you're planning on

connecting to or part time and part of the year or something like that. I just don't want to move back here." And they said, "Neither of those things can— is going to be practical right now. Maybe in a few years you could work remotely but maybe you could be a consultant" so I came back a couple months later and I had changed my mind. I said, "If to work here I need to move back here, I'll do it. I really think what you're doing is really interesting because I think Xerox … they own publishers, they're all about printing, and this is the place I could do my paste-up— copy, paste things." And they said, "Oh, too bad you didn't mention that last time you were here. The company just had a hiring freeze" and so they went through a hiring freeze for quite a while. I— finally the freeze lifted. They notified me; now they knew how to contact me. I applied for a job and they offered me one and that was '72 by that time, and I was insulted by the amount of the offer. I said, "I made this much in my consulting business and just— almost that much, just a tiny amount less, at Stanford University. This— you're a corporation. You're supposed to be paying more" so I turned them down, found out many years later— actually when I was leaving seven years later that no one had ever turned down a job there before and they went on kind of a witch hunt to figure out who it was that had convinced me not to accept the job or had driven me— and it wasn't them. I mean nobody there did that. But Bob Taylor got blamed in the end and for no good reason; I mean it had nothing to do with anything he said or didn't say. So anyway, a year later they approached me again and increased the offer a tiny bit but mainly met one of my requirements, which was— when they were hiring me it was to work in the POLOS project, the PARC Online Office System, which was run by Bill English, a great manager from SRI, one of the inventors of the mouse and so on.

<interruption>

Tesler: So I would have to work in the POLOS group that was run by Bill English who had come from SRI and I liked Bill a lot but the project they were doing wasn't interesting to me. I was more interested in personal computing for consumers and not so much in empowering people in offices, which was what they were working on at PARC Online Office System, and I really was most interested in Alan Kay's project, Smalltalk. So the first year they said, "It can't be done. Alan doesn't have any head count." The second year they said, "Well, Alan still doesn't have any head count but Bill wants you so much that he is willing to say that you can spend a certain amount of time working with Alan's group at the same time as you're helping him because he really needs your experience with making computers easy to use." So I developed that reputation at Stanford and so they'd heard about that I guess. Oh, and also they had used PUB. When I went— when I came back from Oregon and found that I couldn't get a job at PARC right then, Les Earnest, who was the manager of the AI lab, asked me if there was anything that I might want to do for them and he said, "I want to— I'll give you a list and you tell me if any of them are ones you'd be interested in doing." So he listed a bunch of projects that were all over the map, every type of thing you could imagine happening at the AI lab, and one of them was to do this publishing language for generating formatted documents based on something called RUNOFF that was already done at MIT and he wanted to do a way more powerful thing than RUNOFF. He wanted to have a macro language or a programming language that processed strings and you could write your text and you could insert markup just as you did in RUNOFF but it would have a lot more features. It would have tables of contents, indexes, footnotes, cross-references, headers and footers, automatic numbering, subsection numbering, and in fact with a general program language basically it could do anything. So that was the spec and I went "Yes, that's

what I want to do. What are we going to use for our output printer?" and he said, "Well, initially just monospaced type and use the line printer but I'm working on another idea." And it turned out while I was working on what I ended up calling PUB, PUB the document compiler, and PUB was short for publishing, he was working on getting devices that would print proportional spaced fonts anywhere on the page and— 'cause he had a longstanding ambition to get into the computer printing business— computer publishing business, which he did later on. He started a company to do that [Imagen] but at that point he just wanted proof of concept and he wanted something that the researchers at SAIL could use to produce better papers and PhD theses and so on. So I took the language that SAIL had, which was a version of Algol called SAIL, which had an extended character set of characters nobody else had at the time until Unicode came along but we actually had labeled keys on the keyboard and so on and lots of shifting keys to get those characters. And so I just went with it and I used even more characters that were in the character set to have in this macro language and that— it would be called a markup language by— in today's terminology and I got all those features to work in a two-pass process. And the only reason it was two passes is because for cross-references I needed to be able to reference a future page and until it was laid out I wouldn't know what page number it was and while I was at it used that as a way to also make sure that the table of contents had the right page numbers 'cause I got to go back on the second pass and stick those in. And not only did it take off at SAIL and get used a lot by the scientists there; we had just gotten connected to the ARPANET and in an early feat of open-source software we made it available to all the other universities and they grabbed it and the PhDs in computer science at many major universities used PUB for a few years to produce their PhD theses.

Kossow: Then Brian Reid did Scribe based on—

Tesler: Brian Reid had at Carnegie Mellon used PUB and tried to teach it to secretaries and found out that it was too hard. That was a combination of just inherent problems in the design I did 'cause I didn't do it for that set of users. But it also was partly because an acquaintance of mine— or well, non-acquaintance of mine, initially— but a guy showed up one day wanting to talk to me. And he had driven down from Berkeley. And he said that he was developing a markup language based on Runoff for a client of his. He was a consultant. And he had heard about mine. And he was very concerned or the client was very concerned that when they came out with their thing, that it would be kind of squashed by PUB because PUB was going to be more powerful. But he didn’t care about that really. What he really cared about was that it was what he did should be a subset of PUB. And but it was too late in his development for him to change anything. So, he insisted that I make PUB a superset. And therefore, he would help me make it a standard by being like his. There’d already be one other thing out there that had this same syntax and simple commands. And PUB would also have this other stuff that was more complex that he didn’t have. But his was going to be available on more keyboards and things like that, whatever. So, I was reluctant because he had made some terrible design decisions. And I had uniformity all the time. If I had group, I had ungroup. If I had right flush, I had left flush. Symmetrical things using either an “un-“ or an antonym. But he would have things like justify/regular or something. I mean just— or I can’t remember what they were anymore. But they were not related in any way. And that’s why I can’t remember them. They were not intuitive. And so, PUB had a bunch of these. And people always complained about them: “I always have to look it up. I can never remember what the opposite is.” And I’m going, “Argh. I should

never have listened to this guy.” The irony was— his name was Paul Heckel— and the irony was that he ended up writing a book on usability a few years later. And I’m pretty sure it was that experience that got him to be aware of usability. It wasn’t something that was on his or many other people’s minds. But he was—

Kossow: Paul Heckel shows up later in ZoomRacks and in…

Tesler: Yeah.

Kossow: …the HyperCard…

Tesler: That’s right.

Kossow: …lawsuit. So, we’re up to 1974 where you had spent a little bit of time at PARC working in SSL. And you were given an assignment to work with Ginn Publishing.

Tesler: Yes. The first few months that I was at PARC, Bill English let me just kind of brainstorm with another new employee who came from SRI. His name was Jeff Rulifson. And we started just being— trying to be visionary all we could and talk about the kinds of systems that could be there in the future that PARC could contribute to making happen. And I had been thinking about copy and paste, cut and paste for a few years and thinking about modeless editing for a few years. And but here I was finally at a place where I was getting paid for dreaming up ideas instead of writing programs for people to use immediately. And so, I started doing lots of design of possible interfaces. And one theme I kept coming back to was it’s got to be modeless. And maybe it could use cut and paste. Initially, I was thinking it would use cut and paste just for the page layout part of it. And I would have some analogous thing called delete and insert to use for moving text around, but it would have the same concept. After a while, I thought you don’t really need a different name. We can probably teach people what cut and paste means even if they’ve never heard of it before. Anyway, at the same time that was happening, Bill was— Bill English was pushing me to spend more time helping the POLOS guys on their stuff. They were working on a very ambitious system. The main programmers were Bill Duvall, who was building a distributed operating system— and there were not distributed operating systems at that time, so, that was ambitious. And he was doing it kind of all by himself. And I think that was intentional. It was like maybe one guy having it all in his head could do it. And if anybody could do it, he could do it. But it was just too early to be doing that. It was many years before things existed that were like what he was trying to do. And after a while, they realized they’d bit off too much. And they dropped that and they kind of dropped POLOS around then, too. But at that time, that was still going on. And then Smokey Wallace, who later went to Adobe research, and John Melvin were the other two that I worked with a lot. And there were some other people in the group. But they wanted me to be helping. But nobody really ever had a specific thing that was interesting that they wanted me to help with. They wanted me to help with stuff that was really uninteresting, not really my kind

of thing. So, I was pretty unhappy. And Bill liked what I was doing in my visioning, but he was trying to get a product done, basically, that he could show Xerox that we were accomplishing something. Anyway, just as it reached a point where it was getting uncomfortable and I was working on stuff, and I wasn’t interested in it, he called me in one day. And he said, “Ginn & Company was acquired by Xerox a couple years ago. And every division of Xerox has to pay a tax to fund the research center. Ginn wants to come out and see what we’re doing here that is going to benefit them so that it would justify their paying this tax. They think they shouldn’t be taxed because whatever we’re doing must have something to do with making copies in offices and they don’t do that. So, they are proposing— they’re going to come here and propose that we do something specific to publishing. You’re interested in publishing, Larry, aren’t you?” And I went, “Yeah.” “Well, how would you like to work on that?” And I went, “Music to my ears.” And so, at that point, I was kind of an unknown quantity still in terms of building something. So, they said why don’t you work with Dan Swinehart. He’ll work on it part-time, and you’ll work on it full-time. And we’ll make something happen for Ginn. Well, Dan Swinehart, of course, was the person who used to be next office from me in SAIL, who also thought we should get rid of modes, and who I had talked about cut and paste with back at SAIL and still talked about it with him. He was enthusiastic about it. And of all the people in the world you could pick to help me do what I wanted to do the way I wanted to do it, it would be Dan. So, I was delighted. So, I dropped out of POLOS completely, started working on the Alto. Actually that wasn’t immediate. At that time, in theory, we could choose to work on either the Alto or the POLOS. And when Ginn came out and visited, we showed them both. And we showed them where the future path was on both. We were all trying to be objective, but we all had our biases. And they said, “Now we don’t want to go with POLOS. We want to use the Alto,”— for a number of good reasons that they had plus the reasons we all had. So, I started doing that and soon found that I needed to know a lot more about what they needed, how they worked. And they were in Boston area. And I did make a couple of trips there. But I couldn’t really go for an extended period of time, which I thought was necessary. I really needed to be there and see a lot of things that they do that they’re not even conscious that they do. They— even just during the few hours I was there, there was a guy who was telling me he was doing one thing. And he was doing the exact opposite. He was telling me what he was supposed to do. But that’s not what he was doing. He was doing the opposite. But he didn’t want anybody to know that because the “right” thing to do was this other thing. And so, he told me that’s what he did. I thought I really need to be here for a few weeks. But I can’t do that. I’m a single father with a young child. And I couldn’t get away for more than a couple of days at a time. So, not to worry, Bill English [or Ginn] went out there and somehow recruited this young man who was not so tied down. He had a family, too, but wasn’t one where he couldn’t be away for a few days. His wife could take care of the child. And his name was Tim Mott. And I read his resume. And I went, this is completely different from everything else we have here, but it’s actually kind of relevant. He knows a little bit about publishing stuff. And he has worked in— earlier in his career, I think, in a helpdesk situation. And he’s done a little managing and kind of…. He’s not the computer science PhD from MIT kind of person that PARC usually hires but pretty interesting resume. And so, I met him and was just blown away, really. And I thought this is the perfect guy. So— and he’d also done a little bit of fieldwork with users, studying users, which is called ethnography now. But, at that point, it was not called anything. But we soon just started calling it anthropology. Anyway, he went to Ginn for a month or two I think, and came back, moved his family out from the Midwest. And he started working with me. And we designed Gypsy. And we decided we would save a lot of time by taking the Bravo source code, throwing away the user interface, which was completely modal, writing new modules that replaced the

user interface modules. And these new modules were going to be modeless, which turned out to be simpler. A modeless user interface loop, we found by building one, was just a simple single loop around a case statement— a switch statement— saying which event did the user just do. Did they push this key? Did they click the mouse button, whatever? Go do whatever they wanted and then go back to the top of the loop because there’s no modes. In Bravo it was bouncing between the insert module and the command line— the command module, and the other modes— modules— find—

Kossow: So, you just got kind of end up with this big finite state machine.

Tesler: —for search. And so, you’d bounce around to these different modules. And any time you found something that applied to all, then you’d have to replicate it or make some common code they could maybe use. It was very complicated. So, the modeless event loop is very, very simple. And that was actually inspired by LISP, which had what was called a read-eval-print loop, which is basically the same idea. We’d been using LISP. And so, we naturally just gravitated to this event loop model. And so, Tim and I worked together. We decided early on to use cut and copy and paste. We did user testing at every stage of the development. He was the one that came up with the idea of the double click to select a word. We were all— the rest of us were all stumped. By the rest of us, I mean Swinehart, and I, and the other people who would talk to us. Couldn’t think of what would be a good way to select a word. And one day Tim came in and said, “Last night I just was tapping on the table and went tap, tap, tap, tap. How about tap, tap? That would select a word.” So, we got ideas from other people, and generated our own, and came up with this very modeless editor called Gypsy. Delivered it to Ginn, they were just delighted. And— but then something went wrong, which was Tim got interested in office systems, and he switched groups. And I got interested in doing page layout. And nobody was maintaining Gypsy anymore. And it had bugs, or compatibility issues with the operating system or whatever. And the Ginn people started having difficulty with it. And they kept using it, but they were a little disappointed that PARC didn’t dedicate resources to maintain it. And they said— eventually, I think they stopped using it and said they were going to wait until there was a commercial product they could buy to do that and as well as page layout.

Kossow: Were there any books produced using Gypsy?

Tesler: Oh yeah, many. They did education textbooks. And they edited manuscripts. And one thing that was better than a typewriter— well two things were better than a typewriter— one was that it was online. You didn’t have to retype it when the author sent it in if the author used it. But, generally, the author was sending in a typewritten thing. And a secretary would, or typist would transcribe it. And then after it was transcribed, an editor would go through and do markups for copy editing purposes. But a new thing they couldn’t do before, which was indicate where to do bold, italic, and underline because Gypsy could do those and some other things, centering and various other. So, we had a little bit of the markup capability. And it was done interactively by holding down a shifting key, just like the control key today, the command key, and then hitting “b” for bold, “I” for italic. And for underline I *think* we used the underscore instead of the “u” at that time.

Kossow: So— and then I assume that influenced Charles when he did Bravo X?

Tesler: Oh yeah. When Charles Simonyi saw Gypsy, and saw people using it, and saw some analytical studies that were done by Tom Moran, Stu Card, and their team, he could just see that this was better. And so, he started removing as many modes as he could from Bravo while keeping some semblance of compatibility with what was there before for the users who were used to it. And that evolved into Bravo X. And then when he went to Microsoft that evolved into Microsoft Word, which was very modeless like Gypsy, and like LisaWrite and MacWrite and so on.

Kossow: Mm-hmm. So, then from Gypsy— did you do anything between Gypsy and working with Doug Fairbairn on NoteTaker?

Tesler: Yeah. I did a little thing called the IDE. So, I was using Smalltalk a lot. And, in fact, I used Smalltalk to do a prototype of a page makeup system. It was so slow in Smalltalk at that time that we had to have the PARC videographer shoot video of me giving a demo and then play the video back at nine times the speed so that it would— so that people could even follow it let alone me finish the talk in half an hour. So, I developed that and showed people. And it had an interesting feature, which was if you made a selection, it didn’t wait for you to hold down a menu button on the mouse or on the keyboard. It would just immediately pop up a bunch of icons, a glue pot for paste, scissors for cut, and a few other things, bold and italic, and little symbols, and things like that. So, it would immediately pop up the menu super-imposed on the document. And you could then click one of those things to do that command. It was a little bit controversial because what if you wanted to click there for some other purpose, you had to do something to make them go away, click out or something. But it was— it’s something that I thought would be a good idea. Never saw it again until the iPhone. On the iPhone, if you double tap on something to select it, a menu pops up with cut, copy, paste— and— without you first telling it to come up.

Kossow: Oh. Okay. I would say it’s not like a tool palette. It’s—

Tesler: No, it would be a tool palette that would appear just because something was selected. Yeah, there have been some things like that. But this was superimposed and very much like the iPhone, not the original iPhone, but when iPhone added cut and paste. That’s the way they added it. So, I thought it was kind of funny. I was the only person who remembered it probably, unless somebody remembered the demo I gave at PARC at the end of the ‘70s. But it resurfaced in the— what, thirty years later. But the other thing I worked on was that I was very frustrated programming and debugging in Smalltalk because as it grew larger and more powerful, there were just more and more classes and trying to keep track of them, and figure out which class to use, which subclass— other class to use. Debugging was very much a terminal emulator. You’d jump into the terminal emulator, and you would type an expression in. And it would evaluate like in LISP. But there was no way you could see what was on the stack. You had to say print stack. And it would look at the stack. And if you wanted to go to the fourth thing on the stack, you’d have to say display the variables at level four. It was the old style ‘60s and ‘70s interactive debugging and

editing system. So, I thought— this has got to change. And so, I was thinking about that. And one of my colleagues, Diana Merry, was in a meeting one day. And people were talking about ways of searching databases or something. And she said well— and she said this kind of once a month for the whole time I knew her. She said, “Well, as Alan Kay keeps reminding us, no one’s ever figured out a good way to browse. There really should be a good way to browse, but no one’s ever done that.” And I said, “Diana, I’m so tired of hearing that. I’m going to come up with a new way to browse, a better way to browse.” So, I went away thinking, “What would I browse?” Books? There weren’t any— really, books on the network. And— what would I browse? And I thought well we’re programmers. We could browse code. We could browse classes and methods because we’ve got too many of them. We could browse those. So, I thought this is great. Smalltalk’s introspective, we can have code that looks at the Smalltalk data itself. And so, I came up with this idea of having a multi-pane window, which we didn’t have before that. There were no windows with multiple panes. And in a way, it was kind of like an old fashioned screen that had multiple partitions. But this was within a window. And if you— the original version had a class in the upper left pane, and a method in the upper right pane, and then the code of that method in the big pane underneath. So, one was a list— a list of classes, a list of methods, and code. And so, I had to build a new class for multi-pane windows and for lists— displaying lists and selecting lists with list-type selection where you click one and the other ones deselect. And scroll the lists, so they had to have their scrollbars. The editor we already had. Dan Ingalls had already built a cut and paste type of editor. And that— I put that in that bigger pane. And it was the Smalltalk Browser, that’s what we ended up calling the Smalltalk browser, and you could browse code. I did it in five weeks or something and showed a version of it to Dan Ingalls. And he noticed how many classes we had. And he went well, that’s too many classes. I— why don’t we have categories of classes and categories of methods so that you don’t have to scroll through so much? So, I said I’m not sure how you do that. So, he said, “I’ll add it to Smalltalk, and I’ll show you.” So, he added categories to Smalltalk. And once he did that, it was easy for me to build a five-pane browser, which became the classic Smalltalk Browser. And it’s got a lot more power today than it did then, but it’s basically the same thing. It hasn’t changed a lot. Then I thought okay, next I’m going to use the same technology to— instead of having lists of classes and methods— have lists of variables in a debugger. So, first I did something called an inspect window. And you could select any variable that you saw in a code window and say “inspect”. And it would come up and show you all the fields of that object. The other thing you could do is— which I did next, was a debugger, which is that if you hit a breakpoint or there was a crash of some kind, an exception thrown or something or other conditions, it would open up a window that was a debug window. And it showed you your stack. You could click any level of the stack. It would show you all the variables at that level. You click on the variables and it will show you the value. And now that was a three-pane window. So, I had all these different windows. And the last thing I did was a performance measurement tool, performance improvement tool, which was based on earlier things that were done in some systems. Some people called it a spy. And you could have a spy that would sample the program counter and tell you where your— in what leaves of the code you were spending most of your time in. And I thought well, I don’t want to just know what leaves we’re spending our time in. I want to know what branches we’re spending our time in. I want to not only know that we’re spending a lot of time in the print routine. I want to know who called the print routine so many times and who called them so many times, and— or who’s slow and spending so much time doing something. And so, I did a tree-based spy, which is the way it’s done today. Eclipse and other systems do it that way. So, that was the first IDE that was graphically based— based on overlapping windows, and panes, and mouse, and stuff like that.

There were already before that APL, and LISP, and a lot of other terminal oriented IDEs. And even Smalltalk was kind of a hybrid, mostly terminal but a little bit of editing code using a consumer-ish word processor. But that was really the first IDE and was published— a lot of that was published— in the books that PARC did and a big article in Byte that was the big Smalltalk issue— in 1981, I guess.

Kossow: So, that sort of gets us then to NoteTaker, which was the first microprocessor-based Smalltalk—

Tesler: Yeah. That will be a shorter story. Doug Fairbairn— well, Adele Goldberg actually— it was— I can’t remember if it was her idea or Alan’s to do the Alto as an interim Dynabook because they knew the Dynabook was many years away. But the Alto wasn’t satisfying to her either. She couldn’t bring it into a school and have the kids use it. It was way too heavy to move around, and big. So, she wanted something that you could at least— no worse than a suitcase— that you could take into a school, set it up, and have the students use it. Eventually some— you know, be able to take home with them but not yet. Initially, we would just loan them or something. And the most important application she could think of was to take notes. So, they could have it in class, and they could take notes in the class. And so, that was the minimum specification. Of course, we could do anything, we knew. But if it could take notes, we met the minimum specification. That dictated a lot of the hardware features. So, Doug Fairbairn, who works here at the Museum now, volunteered to run the project, and do most of the hardware engineering, and subcontracted out to other engineers in other parts of Xerox, PARC or otherwise. And I can’t remember how I got roped into it. It was probably Doug, but maybe it was Alan, or Adele. “you should really work on this.” And— or maybe I was just curious and asked; I don’t remember how it started— but I ended up working on it. And the Smalltalk part of it was done by a guy named Kim McCall, who we had just hired, pretty young guy. And there wasn’t a lot I could contribute there because we really just wanted a port from the Alto to the NoteTaker with as little change as possible. And so, he asked me questions sometimes. But he basically was on top of it. After the first few weeks, I didn’t have to spend a lot of time with Kim. But Doug was running into trouble. And he was trying— it was going to be too expensive, too big, heavy, etc., etc. One of the issues was that he was trying to use a bit-slice processor to build this— a custom bit-slice processor. And— but he was also going to use a new chip from Intel that was coming out called the 8086. And he was going to use that just for I/O control or something. And so, he was talking about designing this processor, how long it was going to take to design it, and what a shame, and how would we do this, and get it debugged. And I said, “Why don’t we use the 8086 for everything?” And he said, “Well, one 8086 couldn’t handle it.” I said, “Well, more than one 8086 for now until they’re faster.” And so, we ended up with an architecture where there were a number of cards. And each card had a processor on it and had some function, general I/O, Ethernet, main processor board, and a few extras for things we’d think of later. And the 8086 was a little bit away. But when the 8086 was imminent, we were going to get one in a couple of weeks. He said, “Larry, why don’t you design the processor board?” I said, “I’m a software guy.” He said, “You’ve never done any hardware?” I said, “Well I built a joystick with instructions from Byte and made that work on my Commodore PET. So, that’s about it.” And he said, “I’ll help you, but I think you can do this. My theory is any software guy can build hardware. It’s not that hard, especially these days because—“ And then he said, “You have an unfair advantage because we have SIL,” which is a CAD tool developed by Chuck Thacker (who won a Turing award a couple years ago for many things

including that). And we have the Stitchweld machine. I said, “What’s a Stitchweld machine?” He says, “It’s kind of like a wire-wrap tool, but it’s completely automated. And we can sit at the Alto. Do a design in SIL. Push a button. It will transmit it over the Ethernet to the Stitchweld machine that’s sitting in a special air-conditioned room somewhere in the building. And the operator runs it to make sure nothing disastrous happens. But basically it’s all automated. And it does something similar to wire wrapping. And it makes prototype boards really fast. And when we’re building things in low quantity, before we go to a printed circuit board, we’ll just do that. And if you submit it before you go home in the evening, when you come back in the next morning, your board will be wired.” I said, “Well, okay. But how about debugging?” And he said, “Well, we have these fancy new logic analyzers.” And so, he had seen that just at that moment there was technology available, not to most people— except for the analyzer, which was available to anybody— but that plus this other stuff, it was bringing it to a new level of education, which was less education than you used to need to do this. So, sure enough, just in a few days, using some— an example of a processor board that Intel provided, I just kind of adapted that to our bus, I designed the processor board. Sent the design to the stitch weld machine. Plugged it into the NoteTaker. Turned it on. Nothing. Learned how to use the logic analyzer. And after doing that for a bit, I went back to Doug. I said, “Is bit zero the high order bit or the low order bit?” <*laughs*> I had never seen— I mean I’d heard of them, but I’d never seen a little-endian machine before. And— but that wasn’t really the problem. He said, “It’s the low order bit. But that’s— you did that right.” Oh, I don’t know because it’s coming out backwards. It’s like the memory interface chips are getting the flipped signals. And so, we contacted Intel. And we said what’s going on here? And they looked at it and went oh, error in the documentation. Whoever did the documentation thought the zero was high order bit. <*laughs*> The documentation was backwards. So, using SIL, in twenty minutes, I flipped them all around and sent it back through to the Stitchweld machine. And the next day, it worked. The other thing I contributed to was the Ethernet board. We couldn’t figure out a way to get the eighty or a hundred chips that were on an Ethernet board at that time into a board that could only hold like twenty-five chips in the NoteTaker. So, I pulled out Byte— Byte issues that I was avidly reading in those days— and I was flipping through looking for ideas. And I had gotten from [Bob] Metcalfe a list of all the functions that were done and how many chips they took. So, one function that took a lot of chips was a phase-locked loop, and, well, not just chips, but components, a lot of components. And so, I started looking around to see if I could find anything on phase-locked loops. And sure enough in Byte there was an article by somebody that said “software implementation of phase-locked loop”. I went, that’s what I’ll do. But we had a much higher rate than they had, and so it was tricky. I had to do some very tricky coding. But that’s the kind of stuff I love to do in machine language. And got it so that just barely we could fit an Ethernet— three megabit Ethernet— on a card. Somebody else designed the board to my specs. And then I got the software debugged and just barely could keep up with Ethernet. And that was it. The only other thing I did on the NoteTaker was accompany Doug on some sales trips where we towed it around trying to find a Xerox department that was interested in it. And we did, but they couldn’t find the time or the funding to help us. So, the project never went anywhere. And then a guy named Adam Osborne visited PARC one day, was in a conference room. And behind the person he was speaking with across the table was a picture of the NoteTaker. And during the hours that he was there, we can only imagine what went through his head. But a few months later he announced the Osborne 1 that looked just like the NoteTaker.

Kossow: Mm-hmm. So, that— well, let’s see that takes us up to about 1979.

Tesler: Yeah. And— which is when I met the Apple people.

Kossow: Mm-hmm.

Tesler: I had a girlfriend from the early ‘70s, ’73 until about ’79 or something. And in— when I was still with her in 1977, ’78, whatever, she joined Apple, became employee number 32 and then tried incessantly to try to recruit me. But she was becoming my ex-girlfriend, and I really was not interested in being at the same place. And also I just loved what I was doing at Xerox. Her name was Phyllis Cole by the way. We’re still friends. But I did go with her, when we were still together, to the first Apple picnic I think it was, which was at Marine World or something like that, and met a few people, but didn’t feel like Steve Jobs was really approachable. I met a lot of the other executives and senior engineers. But I didn’t meet him. And then a strange thing happened. As I mentioned before, I had a Commodore PET. I also, at the office, had a Processor Technology SOL or some other— no it was an IMSAI. I had an IMSAI. And I was playing around with that. And I was telling everybody who’d come in that personal computers are the future. I know we’ve been saying that, but our personal computers are years away. There’s a lot you can do with these very low-end computers. And Xerox is going to miss the boat unless we get into this. And I had a couple of people who agreed with me, Dave Thornburg and a couple of others. My name— I’m afraid names are slipping me now. But most of the people at PARC ridiculed this stuff, looked down on it, said oh, these hobby computers. They’re not for real people. Ours are for real people that are in real offices, real homes eventually. And I went no, this is moving really fast. I go to Homebrew Computer Club meetings, not all the time, but frequently enough, starting with I think the first Homebrew Computer meeting that was not in somebody’s home. It was at Peninsula School. And I said there’s a lot of people seriously doing stuff. And there’s magazines. And this is really taking off. Xerox headquarters got concerned. And they started sending executives out. What’s going on here? You guys are working on these projects that are taking years and years and years. And you can buy a personal computer. What’s the difference? And we would dutifully explain to them the difference between our powerful, easy to use interactive office system and these, you know, cheap hobby computers. Well, they weren’t convinced. They thought we were— also they thought we were being naïve. So— excuse me, I’ve got to—

Kossow: Yeah. Yeah. You’ve been doing all the talking.

Tesler: A guy named Roy Lahr, I think it was, was in the business development group in East Coast Xerox headquarters. And he was assigned to— and he was a kind of disruptive kind of person, just a little bit obnoxious but not in a terribly obnoxious way, but pushy without being obnoxious, and didn’t like taking no for an answer. And so, he came out to PARC. He said, “We’re going to study— we have a task force— we’re going to study these personal computers. We’re going to come to some decision about what the company should do. And we want your input. But the decision is going to be made by the management of the company.” So, we all gave input. And as he went around meeting people, he found

that there were three or four of us that thought personal computers were what he thought. This was something that was going to happen and that Xerox couldn’t wait until PARC came out with something. And they wrote their findings. And recommended that they, in fact, partner with somebody like Apple to get into this market. And so, they came up with a deal where Xerox would be allowed to invest in Apple. Apple would be allowed to see some PARC technology, which they’ve heard of from the graduate students that they’d hired from various places. And gradually— oh, and PARC would help Apple to do the kind of hardware we’re doing with mice, and bitmap displays, and so on. They would give a limited amount of technical help, and more sourcing help, help them find manufacturers who would make things like the mouse for them. And Xerox’s view was this would be a benefit to Xerox. We’d never have the volumes Apple had. If Apple came up with the mouse, it would probably cost a few dollars instead of the fifty dollars that ours cost to make. And we could buy those from the same place they bought them and sell them to our customers. So, Xerox thought that this was a good thing. Some people at PARC thought it was a terrible thing. It was I think threatening in a lot of ways. But also, we had developed a lot of this technology. We couldn’t really patent it because, at that time, it wasn’t clear what could be patented in computers. And the general belief was that not much except hardware. Software just really couldn’t be. It turned out not to be true, and it changed, as well, legislation. But we had very little protection. And some people were afraid that Apple would steal Xerox’s technology. But nobody wanted to give it away, but we had these limited goals. Let’s get them excited enough that maybe they could manufacture it for us, maybe they could provide us sources where we could buy things cheaper. So, we had a limited idea of what the relationship would be. And Apple also had a limited idea of what it would be. They were sort of they intersected. But they weren’t the same. So, the time came to give the promised demo to Apple. Excuse me. So, the time came to give the promised demo to Apple. And I’ve talked about this many other places. It’s written up. But we first showed them very little. I don’t even think I was much involved in that one. They were dissatisfied. They knew there had to be more. So, we arranged a bigger demo where we’d show them more. We still didn’t show them everything. Nobody wanted to show them everything. We didn’t even want to show them everything that was being done in Smalltalk, or everything that was being done on the Alto, let alone show them other hardware and other systems. But we wanted to show them enough that they would build bit map displays and mice and laser printers and other things that we could get at a consumer-ish kind of price because they were buying it for a bigger market than we were. And that required showing them a little bit more. So, we did. And after that, it just kind of got away. It got away from Xerox. They didn’t really follow up as much as they might have. And Apple ended up getting all this technology, improving on it. And all the main relationship turned into? Xerox Latin America distributed Apple IIs for several years for Apple.

Kossow: Yeah. Do you want to take a break, or—?

Tesler: It’s almost twelve. What’s left on your agenda? A lot?

Kossow: We have all of Apple to go through. We’ve got everything you’ve done after that.

Tesler: Yeah. It’s up to you. It’s up to you. I could keep—

Kossow: I’d really like to spend another two hours.

Tesler: Well, I can’t spend that long. Well, maybe two hours, I guess I could spend two hours.

Kossow: No, another day.

Tesler: Another day, okay. Yeah, another day makes sense.

Kossow: Because we have so much to talk about, just ATG and—

Tesler: I imagine you want less detail than I’m giving you, though.

Kossow: Oh, no.

Tesler: No, okay.

Kossow: This is fine.

Tesler: All right.

Kossow: The nice thing about this has been that everything you’ve talked about you can sort of find bits and pieces, but it’s not all in one place.

Tesler: Yeah.

Kossow: So, this is— four hours would be really good.

Tesler: Okay. I got through ‘60s, ‘70s.

Kossow: Yeah, so you’re just about to go to Apple.

Tesler: We still have the ‘80s.

Kossow: And the ‘90s.

Tesler: Yeah, ‘80s and ‘90s.

Kossow: Yeah.

Tesler: The last decade, there’s not a whole lot to talk about.

Kossow: Well, were you at Amazon when Erich was there?

Tesler: Erich?

Kossow: Ringewald.

Tesler: He was in— he was in a subsidiary or whatever of— he wasn’t in Seattle.

Kossow: Oh, okay.

Tesler: I was in Seattle. He was in Marin County. And I think we did overlap in time, but only ran into each other maybe once or so, or maybe not even while we were both there.

Kossow: Okay.

Tesler: Yeah, he was at Amazon mainly before me, and then ended up with a continued relationship of some sort that I don’t remember very well, but not as an employee.

Kossow: I think that’s where he went after Be [Labs].

Tesler: A Seattle employee. What?

Kossow: I think that’s where he went after Be is went up to Amazon.

Tesler: Yeah, he was pretty early at Amazon. I was several years later.

END OF PART ONE

START OF PART TWO

Kossow: All right. It’s Monday, April the 22nd 2013. I’m Al Kossow. We’re talking to Larry Tesler for a second part of his oral history. So we left off last time with the Jobs demo at PARC. So why don’t we pick it up with you going to Apple.

Tesler: Yeah, I was at PARC and I was— my thinking was very affected by that demo because I was getting better questions from the Apple management than I ever got from the Xerox management. It was clear that they actually understood computers. Xerox was basically still a copier company. The PARC people, of course, were very steeped in computer technology, but not the top executives of the company. So I’d been working with Doug Fairbairn on the NoteTaker computer and I decided one more try to see if we can get Xerox interested in this portable computer. Tried to get an appointment with somebody that everyone recommended us, the last hope and he canceled on me. So I thought, “That’s it. I’m just going to finish up the couple of projects I’m working on and start looking.” So I looked and interviewed at one other company, I think, besides Apple, talked to a couple of others. And then decided that Apple was by far and away the most interesting place for me to be so that’s where I went. And I arrived in July 1980 right after attending the SIGGRAPH Conference and doing a presentation related to the work that was going on at Xerox. And when I first showed up they said, “We’re going to put you in a new group we have, called ‘Research.’” And there are three people in it including you and you’re the more, kind of academic people here in the company and we wanted you to be thinking about what comes next.” So it was Bruce Daniels, who came from MIT where he implemented a popular video game, I think it was Zork. He was one of the team there on Zork. And then there was a numerical analysis guy whose name slipped my mind for the moment, David Hough— David Hough. And me. And that was the— that was the Research Group. So we had maybe one meeting and then at that meeting the first item of business was to talk about whether we should even have a Research Group. And we talked about it a while and said, “You know, there’s just a lot of ideas already out there that Apple could do. We don’t need to be coming up with new ideas. In fact, it would be more fun to work on those things they’re doing.” So we went to the management and said, “We’d like to dissolve the Research Group and we’d all like to join the Lisa project.” And it only took them a microsecond to say yes. They didn’t think we’d want to. So we did that. The first few days I was there had a— had my one and only run in with Steve Jobs, meaning, you know, basically disagreement. And he had told me that I would be in charge of the user interface for the Lisa, or at least the— not maybe in charge of it but the main guy who was going to propose it. “So we need— we need a better user interface. I want you to be the guy who heads up the team to propose it.” Then I found out a day or two later that he had told the same thing to somebody else. I think it was Bill Atkinson that he would be the one. Anyway somebody who one of us ran into said, “That’s funny. I heard the opposite the other day.” So went to Steve’s office, which had a glass wall to all the building— all the building’s offices

had glass walls at least [ph?] three I guess. And knocked on the glass and, you know, “Come in.” Came in and I said, “You blah, blah, blah. *<laughs>* Don’t ever do that to me.” He said, “I apologize,” and he never did it again. And he said, “I want you two to work together. I just didn’t get around to telling you that’s what I really, really wanted was for you two to get together and do it together. But he— and he has just as many ideas to contribute as you do as you’ll see. But I want you to be the leader. You’re the manager, the person with management experience.” So, “Okay. Okay.” So that settled that. From then on we had— we were at peace forever.

<crew talk>

Tesler: I know. Last time I was giving very long answers. So I thought this time I would break and let you give me a prompt.

Kossow: Oh, no. I’m just interested in the whole development side of Lisa, trying to keep it maybe into— I don’t know, maybe 15 minutes is we’re trying—

Tesler: Yeah, well, if you can guide the questions you want to talk about the development of Lisa. You want to talk about, I don’t know, how we got the— I could talk about some of the things I was involved in. There’s the mouse design, the hardware design, doing that with Hovey-Kelley, and then was with Bill working on the user interface guidelines. And, yeah, and the one-button mouse, which is sort of bridges the two.

Kossow: So maybe if you could talk a little bit about—

Tesler: Are you ready, Jim?

<crew talk>

Kossow: So if you could talk a little bit about the development work that you did on Lisa and I guess you were head of applications?

Tesler: Okay. At the beginning when I first arrived at Apple and dissolved the Research Group and went into Lisa, I was an individual contributor. And I decided that I wanted to get involved with the user interface and kind of triage: Which was the biggest issue that would kind of take longest time to fix? So I spent some time looking at the hardware. Was it as easy to swap a board as they said it would be? And it was, yes. And is the keyboard design adequate? It was okay. And the mouse. And so they said, “We’re doing this mouse with an industrial design firm, Hovey-Kelley,” which later dissolved. But David Kelley

went on to found another company and then started IDEO with Bill Moggridge. But in those days it was Dean Hovey and David Kelley. And I didn’t work much with David. I worked mainly with Dean and with an Apple engineer who had been assigned, a mechanical engineer at Apple who had been assigned to work with them. And over I’d say about a three-week period we started with what was a brilliant conception that they had for a mouse. In other words, the basic function of being able to roll around and detect the motion and compute where you are basically, where the cursor should go. And they— their brilliant idea was instead of using a ball bearing like Xerox used they used a loose ball— a loose ball bearing basically. Instead of pressing the wheels up against the ball and trapping it in a confined space, they let it basically just roll on the table and the mouse kind of moved with it. And then the rollers didn’t constrain the ball. And as a result wear was less of an issue because it did— it self-adjusted. And dirt was less of an issue because it would— the ball would tend to, when you lifted the mouse up, it would— it would separate from the rollers and drop out any dirt that was accumulating so it was a much better design. And also way cheaper to build. So I liked that. The problem was that it had two or three buttons. I don’t remember which and I thought that we should start with one button. But a bigger problem was the feel of it to the hand, but they were working on that already. The other thing they were completely ignoring was the cord. And to me the cord was a major issue in mouse design. If the— it’s still true today— if the cord is too loose, which was the cord I think we started with— when I first got there it was basically about the feel of string. It would just go wherever you put it and have no tension to pull it back. And so it would just lie there on the table and you’d move the mouse and you’d run over the cord, basically, invariably, you couldn’t avoid it. It would just— you keep running over the cord. You have to keep pulling the cord out of the way. Then they came in with just the opposite, a very stiff cord. And you put the mouse down and let go and the mouse moves on its own because the cord is trying to straighten out. And so then we tried another one. Then I said, “Couldn’t we just try like 10 cords or something and of different springiness.” And so they said, “Oh, yeah. Definitely. We do that all the time.” And so the next day brought in a bunch of cords and we tried them one at a time until I found one that had the right behavior. Then ran some user studies with that, got feedback from customers. Can’t remember if we had to change it or not, but anyway we had to confirm it, at least. Similarly, the buttons. It wasn’t— the number of buttons was a controversy but the bigger issue was, for me, the endurance, lifetime of the buttons and the pressure necessary. Again, the springiness. How tight was the spring? And how much— how far would you have to push it and how much force would it exert on your finger? And so I had a very interesting conversation with the engineer. I said, you know, “This button is going to be pushed a lot and people are going— their hands are going to tire if it’s too stiff. And the button is going to wear out.” He said, “Oh, don’t worry about it. This button is rated for 40,000 presses.” I said, “Uh… 40,000 presses—I don’t think that’s enough. How many years you think that’s going to be?” He said, “I don’t know. Several years for sure.” And so we did a little calculation on the back of a napkin and I said, you know, “Ten button presses a minute and so and so an hour, eight-hour day, you know, 3 million button presses,” or something like that. He said, “Oh. People aren’t going to press it 10 times a minute.” I said, “How do you think the mouse is used?” He said, “You’re typing. You type for an hour. And then you go take the mouse, and you see a correction, you click that, and you correct it. Then you click there and maybe you correct— make 10 or 15 corrections. So in an hour you clicked it 15 times.” I said, “Have you ever seen anybody use a mouse on a computer?” And so I explained to him what it was really like and he couldn’t believe it. No one had told him what the uses— what the usage cycle was basically. And so he went and he said, “It will cost an extra dollar to have a longer-lasting button.” I said, “You don’t want customers coming in every month to replace their mouse

because the button stopped working.” So they went and they found a vendor who could do it for less than a dollar and did the right thing. So that was my early introduction to working on the mechanical stuff. The other part of it, which was— oh, and also the springiness of the button. You know, it was too tight, then it was too loose. You just rest your finger on the button then it presses. Too tight, you have to really push it hard and muscles get tired. We went through all that and again tried a lot of different buttons and got it to work. But the one button versus two or three was the bigger issue. Most people wanted two. Jeff Raskin wanted one button, but Atkinson kind of liked the idea of one button but he wasn’t convinced it would be enough. And so we ran user studies again and I conducted those in the early days. We brought people in, had them use some very early prototypes that Bill had made of just individual interactions. They weren’t a system yet. Just this demo was pulling down a menu and— or use— operating a menu. They weren’t pull down yet. This demo is moving a window. This demo is about discarding something. And then we’d have people go and try to do it and see whether that was intuitive or what they thought they should do before they did it. And we also had them operate the buttons. And sometimes it was one button, sometimes there were two. And we would have different explanations about what they did in those two cases. One thing we found was that people wanted more function than you could get from one button, but they were willing to use the other hand to shift. If we gave them two buttons that avoided that shift or reduced the amount of shifting, they kept saying, “Which button? Which button? Which button?” And they never could really remember which button did what. A lot of people can’t even tell left from right, so saying left button and right button was a problem. In fact, at Xerox we had a mouse with three colored buttons on it that we used for a while because of that issue. But it didn’t look very good. It was a little gaudy and didn’t work for color-blind people so well. So we went back to left and middle, but at— left, middle, right or left-right. But at Apple we started deciding that, you know, maybe in a few years we’ll go to two buttons when people are very comfortable but we want people to be 100 percent comfortable with a mouse. Has to be just dead simple. And so we thought, “Let’s make it be a one-button mouse. That will be a differentiator.” And besides when people say they want to do something without shifting, well, there’s a lot of things they want to do. And one extra button really isn’t enough on a mouse. You still need shift keys. So why don’t we just have enough shift keys and come up with a consistent meaning for them and that’s what we did. So some people were very opposed. And Tom Malloy, in particular, who was running the word processor group, Lisa Word— LisaWrite, I think it was called. Yeah, LisaWrite. He pleaded with us to have two buttons because he wanted the second button to extend the text selection. And I showed him these experiments we’d run where people would click down on the left button, drag and then let go of the button. And because of that it wasn’t necessary to extend very often and so, when you did have to have it extend, you could use a shift key and it wouldn’t be a big deal. Well, he really couldn’t believe that it was— that the mouse was accurate enough for people to click down, drag, up. And so I showed it to him and they— you know, he saw that it was the case. We did that by having a special heuristic that I developed at Xerox, which was that we kind of— it was little bit like cameras today that automatically control for hand shake. This was a software algorithm that controlled for hand shake as you were dragging the mouse. And it was a pretty simple algorithm. You just looked and saw— well, when the person was dragging horizontally you gave them a wide band of height to wiggle in without assuming they’ve left that line and gone to the next line. Because if you drag horizontally you’re more likely trying to select more characters than more lines— or rows if it’s spreadsheet or something. So as long as they’re starting out with a horizontal motion, we go assume horizontal— don’t you know— but assume they’ll be some shaking in the vertical direction and kind of ignore it unless they go all the way into the next line, like more than halfway into the next line. And

then go, “Uhm.. they really do want to go to this line.” And then flip that to be the new thing that we assume they’re pointing at and then do the same thing on that line. That heuristic worked very well. Most systems today don’t implement it. They have better mice and people are used to mice. They learn mice when they’re two years old now. So it’s not as necessary as it was back then, but at that point it was extremely necessary. The mice were very crude by today’s standards, hard to control. Still he said, “Why can’t we put just a little button for the— for the second button on the mouse?” And so, in the end, I wrote a memo called, “One Button Mouse.” It was talking about what the tests that we had run and the— that Jeff Raskin, and Bill Atkinson, and I all felt that we should have a one button mouse. Gave that to Trip Hawkins who was running product management, copy Steve, and all the others and they went for it. And they went— they loved the simplicity of it. And they said, “This would be a big differentiator.” Interesting thing is that was, let’s see, 1980. And about 30 years later—I’m still friends with Tom Malloy—and one day he said to me, “You know, Larry, you were right about the one button. People just weren’t ready for two.” And I said, “No. No, Tom. You were right about the two button. That’s what people eventually all wanted, not one button.” And we went back and forth about, “No, you were right. You were right.” But it was basically a very close call. You could really have gone either way, but it was— I think it was— symbolized what Apple was beginning to be and becoming, which was: if there’s any choice of put it in or take it out, take it out. Don’t have stuff in there that isn’t 100 percent necessary. It would just make it more complicated or look more complicated. And Apple was already following that philosophy with many other things—on the Apple II and so on, and— but this established the principle that if you can do without it, toss it, as a company philosophy.

Kossow: So did you bring in the process of UI testing or had Apple already been doing some of that?

Tesler: The whole time I was at Apple I thought I had brought it in. I certainly had brought it into the Lisa project. It wasn’t going on at all. But I found out—when I left, Chris Espinosa and I gave a talk on the history of the Apple UI—and when I heard that he was invited, I went, “I guess you’re going to be talking about the Mac UI that you worked on after we did the Lisa?” Because he was in addition to being in charge of publications he was, on the side, the head of the user interface, which was a good idea because the people documenting it ought to make it real simple. They know what’s easy to explain. And he did a very good job of it. He said, “No. No, I’m going to talk about that a little bit. No, I’m also going to talk about the Apple II and the user testing that we did.” I said, “You did what? Oh, you mean the manuals. You used just the manuals.” He said, “No. No. Just not the manuals. I mean we changed the user interface because of things we found. It kind of started out as testing the manuals, but we didn’t just change the manual we changed the software.” And he described the user tests and they were very much like I was doing. So I was corrected, but no one had ever told me that. I knew they had tested manuals but didn’t know that they had actually changed the software as a result. So— but not knowing that when I first got there I said, “We need to run user tests to make a lot of these decisions.” Because people were arguing, “I think it should be this way.” “I think it should be that way.” Even people that both used to work at Xerox, we weren’t allowed to disclose what we had learned at Xerox in terms of the unreleased stuff, which was basically the Xerox Star. We couldn’t talk about that but they had seen Smalltalk, and Bravo, and a few other things. And—but basically people felt like, “We should ignore what Xerox did. They don’t get it anyway and we’re going to do it right.” So we had many debates about things and in the end I would

end up with two or three ways to do something. At first, it was always with Bill. Later, it was gradually with other people, too. Bill would go off and prototype it. We would— we got into a cycle where he would go home at night and prototype. He would come in— he’d get some sleep, a little sleep— come in the next day and hand me the prototype and then have some meetings. And then go back home and go to sleep and get a little more sleep. So— or he would just stay up all night and not sleep at all and then just get some more sleep during the day. And then I would run some studies. And the way I got subjects without the secret of the Lisa getting out was to use Apple employees. Trouble was as soon as they started at Apple, they started using the Apple II, and they started getting— kind of getting contaminated. This computer was for people who had never used an Apple II or anything before. And so I wanted people who were completely unfamiliar with computers. So the way I handled it finally was asking the people at orientation— new employee orientation— to find people who said they’d never used a computer before and sign them up for our user studies. And we were hiring so many people at that time. There were probably 20 new employees every week. They’d find five or six who had never used any kind of computer or terminal before and send them to me. And I’d schedule them hopefully on the second or third day of their being employed before they got contaminated and have them run whatever we were trying out. And it made a lot of difference. We tested many things that didn’t work well that we thought were good and then finally landed on the design we did. Bill started writing a document called, “The Lisa User Interface Guidelines.” When he came to issues he brought them to me, we’d talk about it, and go through that process. But he kind of used it as a way to drive, “Are we covering everything? If it’s in the— this should be in the spec and there’s nothing I can write about it yet. We need a UI for that.” Then we’d work on that. That turned into a structure where, after a couple of months of my being there, we spun the Lisa project out of the regular engineering group into it’s own group that was more like a business unit with a marketing— product marketing group, which was really more product management— and hardware and software engineering. And I was put in charge of managing application software. There was another guy managing system software and that person changed— operating system software— and that changed a few times— who was doing that— I think we had three different people. And then the hardware engineering group, which was, I think, Rich Page ran that ,or was at least their senior engineer when he was no longer running it. And then there were still arguments about the UI, so they formed a three-person group, one person from product management or product marketing named Dave— David O’Connor, and Bill Atkinson, and I. So engineering had a two-to-one advantage and— but we really just worked on issues together. David would raise a lot of things we hadn’t thought of that the customers are going to want to do. Some of them were in the market requirements documents, some of them weren’t, but he was obviously right. And then he was very good at taking our decisions and selling them to the marketing group. And then my job was to sell it to the engineering group because they had more work to do and Bill was the one who usually demoed it to Steve, got his buy-in, and then he came back with a little new idea or something else to try.

Kossow: And so one thing I was just thinking about was the way that the Lisa system was structured. That it had, as opposed to the Star where everything is sort of all rolled up into one big thing, Lisa had separate applications. So did that fall out from a marketing requirement that you wanted it to be able to do this set of things so you just divvied up between a bunch of people, or …?

Tesler: Well, I think there were a lot of reasons. First of all, we weren’t allowed to talk about the Star. And secondly, I don’t think a lot of us really knew what the architecture of the Star was. We were working in PARC pretty much, not at the SDD, System Development Division, that developed the Star. But it was clearly more integrated. I’d given a demo of it— not at SIGGRAPH. they wouldn’t let me do that— but internal to Apple just before I left. So I was comparing the Star with the things that went before it: Gypsy, Bravo, and things that even preceded PARC. But it was— there’s Conway’s Law, which says that the structure of a computer system reflects the structure of the organization that built it, roughly it says that. And that was Mel Conway, who became a Mac developer in the long run. He designed and help to develop Lightspeed Pascal. But at that time, it wasn’t that well known of a proposition, but it was in partly the way we were organized into different groups. But, no, it was the idea that there would be applications that would be developed by Apple. Eventually, there would be third-party applications but Lisa wasn’t designed to not have them. It was designed to have them. We just didn’t want to take the time to support third parties yet because we were still in the middle of designing it, and making changes constantly. It would drive them nuts. More likely, it would make us freeze it before it was really ready. So it was more about that. That we wanted third-party applications, but we needed a way for them to talk to each other. In fact, the whole definition of the Lisa was summarized by John Couch in an elevator pitch, basically. He took the market requirements document that was written by Glenn Edens and a couple of other people— Barry Margerum, I think, and Trip Hawkins I think it was— I think they were the three names on it. And what John Couch did when he took over the division was to say—every time he came to see us it was the reminder— he said, “Remember, what I want is I want to put the numbers into a spreadsheet. I want to make a copy of the table that’s in the spreadsheet, put it into the word processor document, and then print it and it all comes out together, mixed together text and stuff I’ve printed. And if I go to the graphics program and I create a graph, I want to put that in the Word processor document too.” That’s it. He said, “Make me something that does that and I can sell it,” basically. So we had to keep reminding the engineers: that’s the goal. The main fact— the main thing about this— the main differentiator is this integration. We eventually adopted cut and paste shortly after I got there because that’s what I convinced people they should adopt. And so the story changed to, “I want to be able to copy it from the spreadsheet, paste it into the Word processor document, generate a graph in the spreadsheet, copy that and paste it. Or draw my own picture in a graphics program and paste it,” and so on. We kind of ignored that for quite a while and then one day Bruce Daniels who was the— really the senior architect of the Lisa operating system, he kind of intentionally put it off because he didn’t want to constrain what people did. And finally we got to a point where people were bugging him about it. And he said, “Okay. I think it’s time. We’re going to define the way we can share data.” And so he looked at all the apps at the stage they were. It was about a year before it was supposed to— a year before it was announced, I guess, something like that, oh, a little more than a year before it shipped. He said, “We better hustle. We got to do this. And I want to make it easy for any application to comply.” So we had a user interface that had copy into a clipboard, paste from the clipboard into the document. And so that gave him an idea of how to do it. He said, “Well, we’ll have a storage area in memory. If it doesn’t fit in memory we’ll put it on a disk. And since cut and paste— at the time you cut you don’t know what kind of document you’re going to paste it into we’ll put every format that every possible receptor of the data could possibly want as is known by the cut— cutting application.” So if the cut-copy application knows that graphs are sometimes just bitmaps and sometimes they’re vector, sometimes the text is again bitmap or vector. Sometimes it’s got fonts. Sometimes it’s plain text. They’ll put all that into the clipboard buffer and then the receiving application

can ask for the most sophisticated thing it can handle, the richer text— the richest text it can handle and then ask for less and less until it gets something that it wants. And the minimum for any application is going to be plain text and if it’s a graphic even, it could be plain text that’s just a caption saying what the graphic was. But a little better than that would be just a bitmap of the graphic. At that time, it was all black and white and so that was simple bitmap— or maybe it was grayscale but it was no color— in the Lisa. So in fact, I think it was a one bit per pixel, the machine. So that’s what he came up with and we negotiated with the various groups. And they had to make some changes to their software to make that work. Mostly people objected to the two steps and they wanted to review this whole idea of using cut and paste. “It was a bad idea. We should go back to a different model where you— first you say move. We say where it’s coming from, where it’s going to, and then things happen.” But we were opposed to that because one thing that was very important was to be able to copy from one floppy disk to another. And so you have a floppy disk and with copy and paste you could make a— select it, make a copy, eject the disk, put a new disk in, open another document, and paste. And first you had to say, “Move,” and then do all that stuff. You’d have to go back and put the first disk in again now that you know what you really want. And everybody said, “Oh. Yeah. Never thought of that.” And so because of disk swapping, and today it would be thumb drive swapping, there’s a definite benefit of having that— no interruption between those steps of missing media. And so— but if you use cut and paste you don’t have to worry about that. So that’s why we ended up with that and it’s still being used today. And sometimes for the same reason. iPhone uses it. It’s really when the first iPhone came out, it only really ran one application at a time for all intents— for typical purposes and had the same problem. So it had to be two steps, copy it now, switch applications. The first application— the operating systems are trying to take away all its resources and shut it down, and you don’t want to be in the middle of a copy and paste when that happens.

Kossow: Okay. Let’s see how we’re doing on time. I turned my watch off and my clock off.

Tesler: You’re shooting for 3:00 or something?

Kossow: I think so. It says about 20 after 1:30.

Tesler: It’s 2:00 now. We started— we didn’t start to 1:30. I got here late, 1:15. yeah.

Kossow: Because there’s all this stuff I want to cover. So as you were talking about the requirements, one of the things that I noticed was missing was networking.

Tesler: Good catch. Steve Jobs wasn’t that interested in networking when we at PARC demoed the network to him, Ethernet local-area networking, it just didn’t relate to anything that he thought was important and he was so blown away by what he was seeing in the graphical user interface, he didn’t really notice the networking. And he also didn’t notice or pay a lot of attention to our programming environment. We gave him a Smalltalk demo, object-oriented programming, showing how we could make

changes and get instant testing of the changes. We were very proud of that, being the Smalltalk group doing most of the demos. And Bill was sort of interested in that. Some of the programmers were. But more curiosity than compulsion. Steve was just really focused on the UI, and the way it looked, and after the simplicity, and the beauty of it. So it just wasn’t in the spec. The other thing was: there weren’t really any examples of networks at that time that were a clear-cut win for where Apple wanted to go. The ARPANET was restricted to academia and places with government contracts and so on. Local-area networking was pretty primitive at that time. Ethernet— Apple was— I mean, Xerox was just beginning to work on getting Ethernet out there as a public standard. There weren’t a lot of applications of networking. So what we were biting off was a big enough piece. What we did have to do though was printing because remember John Couch said, “I have to print it when I’m done.” He didn’t say we had to scan anything but he said we had to print. So we needed a print architecture and we did have an effort to develop that. That was Steve Capps and Owen Densmore and maybe Jeff Parrish— there was a third person. They worked on the page setup dialogs, and print dialogs, and how you would handle devices that were—believe it or not— had character imprints on them like typewriters, chain printers, things like that versus bitmap- and pixmap-oriented printers, dot matrix as well as laser. So that was all in the planning and that took a lot of effort. But taking on networking at that stage would have been too much. Even if we knew it was important, I don’t think we would have done it.

Kossow: The development environment sort of leads us then into Object Pascal and eventually the development of— I’m blanking on.

Tesler: MacApp?

Kossow: MacApp.

Tesler: Okay. So when we finished the Lisa and got a lot of good press at first, except- except for the price, which was, I think, ten or twelve thousand dollars, whatever it was, for what was supposed to be a lot cheaper and a lot smaller. But because we were under a lot of time crunch and behind schedule basically, the engineers— the engineering managers’ solution was to just get code working, not spend any time tightening code. Traditionally, back when computers were tiny like that, programmers would write code, get it working, and then spend a lot of time not only getting rid of bugs but shrinking it so it would fit into their, you know, 32K Apple II, or their, you know, 2K IBM 650 or whatever they had. And the Lisa group acted like they were using a big timesharing computer back at their university or something and they just wrote code like there was virtual memory and everything else going on. And we didn’t have that. So it was puzzling why it was so big and I took a look at the code and went, “Oh, my God. You guys haven’t done anything to consolidate code. Here’s like five pages of code that could be done in a half page. Why don’t you take a couple of days and do that?” “We can’t. If we did that for everything in the system and really got it small it would add six months to the schedule.” Went to marketing and they went, “Oh. Trade off of”— Sorry I have the mike. Shouldn’t do that. Went to marketing and they said, “Oh. A trade off between schedule and cost. We just got to get it out. We’ll get the cost down later. Promise that

you’ll work on the code after we get it smaller.” Little did they know that all those engineers would be laid off. But we went ahead with it being too big and too expensive and it bombed in the market. The Mac came out a year later, but in the meantime, in an attempt to save the Lisa during the first year, when it wasn’t ready to be killed yet because we needed a substitute really before we killed it, had to have a Mac. Marketing, I guess, came in and said, “We’ve been talking to customers, and they say there has to be a way to develop third-party applications for this. The IBM PC, which was just coming out has a way to develop applications and the Apple II has a way to develop applications, and the Apple III. We need to have this develop applications.” And I said, “At the beginning you said definitely not necessary. That there would be six applications developed by Apple. They would do everything that an office worker would ever want and having applications would only be a complication. You wanted eventually to have applications, you know, for universities to play with it or something new that came along or for our own engineers to build. But you said that it was not going to be a big third-party program. This was about, you know, kind of an office appliance.” And they said, “Well, no one’s going to buy it. It has to have applications.” So I put together a group and we started building what we called the Lisa Toolkit. It was an object-oriented environment. We modified the Pascal compiler to handle an extension that had classes of objects and started building a very simple framework, the Lisa Toolkit. It had windows and frames, which were things— parts of windows— and had documents, and other things that were similar to the sort of thing we would have done— and kind of learned to do with Xerox. But we didn’t have a model-view-controller interface. The document was kind of like a model. The window was kind of like a view, but maybe the frame was more like a view. And maybe we had— I guess we had views. That’s right. We had views and they went in frames, and the frames went in windows. We had no controllers. I never understood controllers when I was at Xerox and so I didn’t— I couldn’t even explain what they did. And I just felt it was not the right model. Instead I wanted something more like the way the end-user thought about it. So Larry Rosenstein, who was on the team, went off for a weekend and came back and said, “Here’s an idea. We have a command be an object.” I said, “A command? It’s verb. It’s not an object.” He said, “Listen. Hear me out. We make a command be an object and then the object responds to a few messages: “do”, “undo” and “redo”. And we get— this is the way we implement undo. Every object— every command knows how to undo itself.” Because we had undo on the list of things we had to implement. And we— we’re just going to let the programmer do it any way they wanted. But he’s saying this way we can have a standard way to do it. We can have some protocols about it. It’s a brilliant idea. So we adopted command objects. And a lot of people still use those today. It’s not the only way to do it, but a lot of people still use those. And we had some other innovative things in there and one of the things we did was have a certain development methodology that we didn’t have a name for. It just felt right and this started a little bit on the Lisa Toolkit, but when we got to the MacApp, which comes— which I’ll get to in a moment, it became more important. We built— it was just kind of serendipitous. We said, “Let’s have something called the generic application. It does nothing but it does everything the standard way.” It has the file menu, and the edit menu, and it says cut, copy, paste but there’s nothing to cut and there’s nothing to paste. And there’s no files to save. It has a window menu if we want a window menu. It has a drag— if you do have any text it has— it knows how to handle dragging through text. It knows that undo can be undone by redo. It’s very generic and it has an event loop that handles the inputs of the user and hands them off to— a so-far-nonexistent place, etcetera. So let’s teach people there’s this generic application and then they modify it a little bit at a time and turn it into their application. So we said, “Well, let’s do that ourselves.” So we built some classes for the library basically, a framework, and pretty small.

And we built the generic application, the very first one, and it could just do a couple of things because we didn’t have much library yet. And we got it working. And we said, “Let’s make this really solid because we don’t want it to crash during a demo.” So we made it really solid and we showed it to people. And then we got feedback and then we fixed the problems. And then we said, “Okay. Now, let’s add whatever the next thing, scrolling. Let’s add scrolling.” Or, “Let’s add editing. Now, let’s add printing.” And one thing at a time we just added and then we got it to be really solid and so on. So we came up with this thing that was turning into like one-week cycles of development. And we kept doing this for the MacApp project, the Macintosh version of it. And in the meantime we had hired Dan Ingalls from Xerox to work on Smalltalk and he put together a Smalltalk team. And they were very curious what we were doing with object-oriented programming and saw the methodology we were using. And they went, “Yeah, we like that. You know, we like that. That’s kind of the way they do their own programming. “Let’s try to put this into our Smalltalk group.” And it was actually very much like the Smalltalk group at Xerox did. They would incrementally build things, get them working, and so on. And so it turned into a kind of early form of scrum or agile kind of programming. And Kent Beck was in that group and he was very hot about these ideas and took it from where we had done it and left off— where we had left off and took it— took it from there. The other thing we did was that we decided that a lot of bugs were due to the fact that people would allocate memory. And since there was no garbage collector, they had to then de-allocate the memory. Or they would put up an alert box, listen to the interaction, and then take away the alert box. There were things that you would do in the software that you had to kind of reverse later. That was not undo in the literal sense, but you would open something later, you had to close it. And— or allocate something then you had to de-allocate it. And we kept looking for a way to avoid that bug and we came up with a way to do that, which was you always put the allocating routine and the de-allocating routine in the same class, on the same page. So at any time you edit something that calls the allocator you also— do a corresponding edit in the thing that calls the de-allocator or the same thing for dialog boxes— that anything that does something, its reverse should be its neighbor. And so we started documenting these things in the— by that time the MacApp manual. And they were basically what we call patterns today. It was— it was a pattern that you would put the allocator and then the de-allocator in every class that did that. So we went, “Uhm.. this is interesting. All programming should be done this way.” So that was another interesting thing that we came up with and showed to Kent Beck and many other people. So I thought that was— that was a pretty good project. Lisa Toolkit only got used once. Ironically, it was by a company called Compugraphic who built a desktop publishing program with it. Just as they announced it— like days after they announced it— Apple announced that the Lisa was going to be terminated and first gave them a courtesy call that— and told them it would be terminated, and they had just developed this application. The only application for the Lisa that had any development done on it by a third party. And they were pretty unhappy about it. And were furious at Apple and we then said, “But you could develop it for the Mac instead. The Mac is going to be a much bigger seller.” And they said, “No way. We did all this work. The Lisa has a bigger screen. Not interested in the Mac. Won’t work for desktop publishing,” and they canceled the project and let it go. Well, a couple of people who had seen their demo at the conference that they introduced it at just before we canceled it, thought it was a fine idea. And they could do it on a Mac. And so desktop publishing first appeared on the Mac as a commercial thing, even though it was first demoed on the Lisa, that never went to market. Now, in terms of how we got to MacApp, there was a salesman, an Apple sales force— the Lisa sales force— who learned how to use MacApp. He was an engineer, too. I’m sorry. He learned how to use the Lisa Toolkit, selling Lisas. And he

was using it as a way to get— to show companies how they could develop their own applications. And doing Office-sell basically. And then he started developing something similar for the Mac when the Lisa Toolkit was canceled. Well, we didn’t know about him but Larry Rosenstein and Barry Haines and I and whoever else we had at that point thought the same idea. Well, you know, we should maybe do a little— tune up a little prototype for the Mac. And so we made something just super simple. It was like just past generic application that could do one thing. And it was just for our own entertainment and we would go maybe show this to the management or something. One day— Bud Tribble came in— he was my manager at the time on the Mac project— and he said, “Uhm.. I hear you have the Lisa Toolkit kind of moved over to the Mac. Can I see your demo?” I said, “Well, it’s not much there.” He said, “Just show me. Show what you have.” So I gave him a quick demo. He said, “Thank you.” I said, “What’s going on?” He said, “Well, there was this sales guy at this demo we gave to the sales force in Hawaii and he had a little booth and he was showing off his own thing. And Steve was pretty interested in it but he was kind of upset that the engineers here at Apple hadn’t done it. And I told him that I thought you had told me one day that you were working on something like that. And Steve said he wanted to see it. But he sent me first to make sure that it was correct.” So a few minutes later Steve walked in. He said, “Bud said you have something. Show me.” So I showed it to him and, again, did hardly anything. But I showed how you can— how quickly you could add another function to it. I said, “Let’s add so and so to it,” which is very much like the demo we had shown him back at Xerox with Smalltalk. This was a little bit slower because it wasn’t interpreted. It was compiled. But he got all excited. He said, “You’re sitting on a gold mine. Why aren’t you doing something with this, blah, blah, blah?” And I said, “Well, we really don’t have the resources to do this. This was just a demo to ask for the resources.” He said, “You got them.” And so suddenly this was an official project and we went and that turned into the MacApp project. MacApp was used by a few companies, not the hundreds we expected. The Apple evangelists were never comfortable with MacApp. They didn’t really understand object-oriented programming at that time. And they didn’t really want to take the time to learn it because they were very busy selling the Mac and they had figured out how to convince developers to use the Mac. They complained constantly about stuff but they could do anything they wanted with the Lisa— I mean with the Mac Toolbox. The Lisa Toolkit had constraints at that point and so did the MacApp have constraints. By the time MacApp got strong enough that kind of anybody could use it, too many people were familiar with the non-object-oriented way to do it. And it took many years to swing the developer base over to object-oriented programming. It took C++ and it’s popularity and then it took NeXT coming in with Objective-C. But there was one application done in MacApp that was so important that you couldn’t cancel the project and that was Adobe Photoshop. It was implemented in MacApp. And I guess the guys who did it knew object-oriented programming from some other way and they didn’t want to use anything but. And it was also very appropriate for Photoshop. And the more versions of Photoshop were done by different programmers who didn’t know each other’s code, the less they could afford to start over and do it from scratch. So MacApp lasted many years past any lifetime it should have had.

Kossow: So this was in 1984, 1985?

Tesler: The development of Lisa Toolkit— Lisa came out in January ’83. Even before it shipped we started working on the Lisa Toolkit. In fact, I started working on the Lisa Toolkit in October ’82. As soon as

we— where it got into the final bug fixing and manual tweaking phase, the QA phase of the Lisa, beta testing basically. We were faced with this issue, about , developers want a way to build applications. And so I was pretty burned out. I said, “We just hired Eric Harslem. He can manage the Applications Group. I can work on this other thing instead and <chiming of iPhone followed by muttering> We had just hired Eric Harslem, I think from— maybe, I think from Xerox. Although he may have worked somewhere in between. I don’t know. And he was on a management track and we immediately put him in my job. And then, oh, in fact, I think he put— he may have put somebody else in charge of Applications and taken over all of Software. Something like that. But anyway I was free to go work on this Lisa Toolkit idea with a small team starting late ’82. We announced it in the middle of ’83, Compugraphic’s— Write ‘n Set they called it— was done right after the Mac was announced in January ’84. Lisa Toolkit was probably canceled maybe April ’84. We started working on this thing that became MacApp and it became official, I’m guessing, Summer ’84, official project, something like that.

Kossow: So about that, I guess, it was 1985 the Educational Research Group became the Advanced Technology Group?

Tesler: When the Lisa and Mac merged in, I guess, April ’84 or something there was this horrible meeting where Steve came in and announced that we were going to classify everybody on the Lisa team into A, B, and C players. Give the A Players really good jobs. Maybe keep some of the B Players and get rid of all the C Players. That was probably the most demoralizing meeting I was ever in. And but that is what we did and there were people who were considered really good who had no place in the Mac Group or in the merged group but wanted to keep them. One was Wayne Rosing for example. And Wayne said, “That’s okay. Really interested in education. I’m really interested in research. I’ll start an Education Research Group. That’s a very important market for Apple. And I can do— you know, not just K-12 but university level computing we’ll work on that here.” And he put together a pretty interesting group. A lot of people he drew from Lisa. He brought some people in from outside and started this pretty cool research group. I was barely aware of it. One day I got a call saying that there’s a guy who applied for a job there and he won’t take “no” for an answer but he’s not really qualified. And he wants to talk to you, would you talk to him. So the guy came in. His name was Steve Perlman and he had worked at— not Atari— the other.

Kossow: Coleco.

Tesler: Uh...

Kossow: It was Coleco.

Tesler: Coleco. Yes, I think that’s right. And he wanted to work at Apple. And I don’t know whether it started out that way but he ended up wanting to be in the Education Research Group. Anyway, whoever it

was that interviewed him wasn’t hiring him and he thought they were making a big mistake. And so he somehow knew my name. Came, found me, talked to me. So I kind of interviewed him and went, “Yeah, why? I could see why maybe personality might turn him off a little bit but this guy is good, you know.” So I called back and said, “You really should hire this guy. And if you don’t, I’m going to find a way to hire him.” And he ended up— I don’t know if that was the original place that called me or the second thing they offered him but— the second thing he looked at— but he ended up in this Education Research Group. That was kind of the first I became very aware of it. After another re-org, which was the re-org when Steve left— Ed Birss ended up— well, they divided engineering up among various people and there was some part of engineering that was divided—software engineering that was divided up between Ed Tribble and— I’m sorry, Bud Tribble and Ed Birss. And I was reporting to Ed Birss. And initially that was this what be— what had been the Education Research Group and became the ADG Advanced Development Group. And so I was a little annoyed because I thought Ed was going to be doing just maybe the ERG and I was going to be doing the ADG but I was out of town on a trip and this happened while I was gone. Never be out of town during re-org. They had assured me that they would get me the job I wanted but when that— when they saw the shape of the organization that wasn’t going to work and so they came up with this other thing. But Ed said, “I’m only going to be here on this job for a while. It’s a stepping stone. And then I’ll recommend you for the job.” And that is what happened. He only was there for a few months. He became the head of Mac Software and recommended me for the job and I got that. At the same time, we were wanting to do more. The management wanted new ideas. Steve was gone. Where are the new ideas going to come from now that we got rid of Steve? We don’t have— we’re going to run out of Xerox ideas. Where are we going to get ideas from? And so they decided they needed what they were calling an Advanced Technology Group. It was really an R&D group, a lab, you know. And so they came to me and they said, “We want to bring in somebody to run this big lab.” So we started interviewing people and we brought in some really impressive people but some of them just didn’t want a job like that at a place like Apple. And other ones just didn’t get through interviews very well. They just— something about their style, it wasn’t a fit. And we ended up not getting anywhere for a while. One day Del Yocam called me in and I think I was reporting to him at the time. And he said, “We decided that you’d be just as good as any of these people and we want to give the job to you. So you’re now in charge of Advanced Technology.” And that was the beginning of that. So suddenly I was— I’d been a director for only like four months or six months or something and suddenly I was a VP and put in charge of the group.

Kossow: How big was it when you took over?

Tesler: Let’s see. When I first took over ADG it was I think 12 people. And I remember the number 32. I think that was the number that was in ATG on the first day, but it didn’t stay that way long. People started coming to me. Somebody came to me saying, “We have this group called Engineering Computer Services that does support of all the engineers computers and we’re all too busy getting products out to pay attention to this group.” In fact, I think it was the manager of the group came to me and said, “I’m not getting any attention or priority. You’re not so wrapped up in day-to-day development. Could you manage us?” Then Rod- Rod something, can’t remember his last name. And then Monica Ertel came in and said, “I’m running the Apple Library,” and same story. “Nobody’s giving— nobody is paying attention to us.” Or it might have been Ed Birss came in and said, “Monica is running the Library and I don’t have time for

this.” So one by one I started acquiring groups and over a period of months it was suddenly 80 or 100 or something people and grew from there up to over 200 at one point. At that point, we were operating the Cray and we had graphics— a big graphics group that we’d hired a bunch of people from SIGGRAPH conference one- one year. And we were doing stuff for— stuff for the Newton eventually, then some.

Kossow: Right. So was Gassée coming up with all of these ideas then that he was funding little groups for?

Tesler: No. The ideas came from different places. There were a few things that Gassée kicked off. One was at one stage— I think it was after Ed Birss— there were a series of managers. I don’t remember the order off hand. I could reconstruct it probably. But there was Ed Birss. There was Sam Holland running engineering. And there was Steve Sakoman. And they- they all had relatively short tenures because they all really wanted the job as the stepping stone to something else. And Sam Holland who was in his job a very short time and started pitching this supercomputer-power machine. And he knew exactly who to hire and, boy, he could manage them himself. He would— he would even give up this high status job he had managing all of engineering or whatever he was managing to do this project with his— with the little team he could put together. He knew exactly who to hire. And he got Gassée very excited about it and so he— but Sam said he wanted the Cray to use in the design of this supercomputer-power personal computer. And so we found some other uses for the Cray that justified buying it— for mechanical engineering simulations— and so we bought a Cray, $15 million I think it was. Anyway that was all going on not in ATG. And but kind of parallel to ATG. And at some point Gassée came to me and said, “Well, Sam started this thing and he’s hired, I think, two people now. It’s starting to become a distraction. Somebody else is going to be taking over the group. And he’s got to devote full time to it. Maybe he could report to you.” And so Sam started reporting to me. At that point, I think he had four people and then grew it from there. And it kind of was connected with the Engineering Computer Services because the Cray was connected to that group and they all— I think they all sat in one building because he needed, you know, his Cray terminals or something.

Kossow: When did the Apple Fellows start?

Tesler: Okay.

Kossow: Because I know you managed the Apple Fellows as well.

Tesler: Yeah, I think it was just about when we started ATG as a group, very shortly after that. One of the things that grew was that they started the Apple Fellow Program. That was all part of the same, “We have to retain our innovators— before Steve could talk anybody into doing anything for no money at all. And we need— we need some incentives for people to stay. We have a lot of people that are the top rung of the engineering ladder. We need more rungs in the ladder. And we ought to have something that’s—

kind of like the IBM Fellow Program. It’s something that is a very high title, the level of a Vice President and that would attract and retain the very best people.” And the initial definition of it was that it had to be somebody who had made a strong contribution to Apple and a strong contribution to the industry. It wouldn’t be just something that was well known inside Apple— but everybody knew about this thing. Over the years, we kind of modified that that it was okay to have, you know, even kind of one or the other. But the main thing was that it would be a big impact on the industry. And— oh, I remember why it was not so important to be a big impact on Apple. It could have been done before they even came to Apple. And that rose as an issue right away because the first two Apple Fellows— the first three were Steve Wozniak, obviously, Bill Atkinson was also a super choice, and Rich Page who had done the hardware on the Lisa and was a little more controversial because he hadn’t really worked on the Mac. But the management recognized that he had— that he was very strong and we didn’t want to lose him. We lost him anyway. He didn’t stay very long. But Al Alcorn was being recruited and he had done Pong that was— he had done a lot of other things. That was the famous thing he had done. And they also wanted to recruit Alan Kay and he had done— what am I— what do I have to say about Alan. I won’t take time here. But, you know, he had done at other places. But they wanted him to come to Apple. They wanted them both to come to Apple. So we added— we brought them both in, made them both Fellows, and redefined the definition so you could have done your great work elsewhere and then come here. It was pretty controversial at the time. A lot of people objected, not to Alan and Al coming but to making them fellows. And then the program expanded and it even included a person who was not an engineer. But had— was an innovator, Kristina Hooper. I believe she was a Fellow. I may be wrong about that. She could have been a distinguished technologist or something. We had some other name, but I think she was a fellow. But there were few and far in between. It was a rare title to have. [She was a Distinguished Scientist, not a Fellow.]

Kossow: So we’re kind of in the late ‘80s. So there’s just all this money coming in. Apple’s got great margins. They’re all these crazy projects going on in ATG. And then the ‘90s hit.

Tesler: Yeah. Well, ATG was kind of running like clockwork. I had a certain philosophy behind it. And a part of that was that— and this was based on actual experiments over the years— that every year, 15 percent of the people in ATG would leave ATG and go to Product Development and take their ideas with them; join forces with other engineers who were sometimes more development-oriented than they, but some of the people in ATG were product developers. And they’d go in and they’d turn their ideas into products. And then we would take that same number of people out of the Product Engineering Group, which was about 10 times bigger than ATG. So that would be maybe two percent of their engineers and bring them into ATG. And that way ATG stays the same size. We also had a growth budget but in addition to the growth we did by hiring people from the outside we swapped, basically, people with— people from engineering. And it tended to be engineers who were burning out. They had five projects in a row that had no space between them and they were long hours and they just wanted a breather. They wanted to try out some ideas they’ve been thinking of for years and never had time for. And so it was a very strong connection. It was based on Gordon Bell’s observation that the only way to transfer technology is to transfer people. And I was sold on that the minute I heard it and we were doing that. There were people who were against what we were doing. We were starting projects in ATG and then moving them to Product Development. And some of the Product Development managers were idea people and they didn’t

like the i— the fact that before they even got a chance to come up with their way of doing multimedia we were coming up with a way to do multimedia and sending it to them half built and ready to productize. We thought it was great. And so did Product Marketing thought that was great also— Product Management. But the engineering managers and some of the engineers in Product Development felt like it wasn’t fair. They were doing the hard work. We were doing, you know, fun work and we were getting to define the next generation of stuff and it wasn’t right. So we had opposition. And in 1990, I transferred over to the Newton project to see if I could save it and turn it into something a little more practical than what they had been doing. And turned it [ATG] over to David Nagel and at the time I turned it over he had been one of my direct reports. The Engineering Management team basically had made a proviso in there somewhere that we were going to review this policy of this 15 percent and this kind of what they felt was a monopoly of new ideas having to come from ATG. And sure enough, right away that changed. I can’t even characterize how it changed. You’d have to interview David some day. But it changed and the main thing I remember is that they decided that ATG should have a longer horizon. That we were operating with a horizon that was— we called it 18 months to 10 years and— or five years, I think, we said— 18 months to five years. If this could be turned into a product in 18 months to five years then you could work on it, credible path to product, we called it. And they wanted to move that 18 months further out and make it be more like three years, five years, you know, and like five to ten years. Between David and the people who followed him, Don Norman and Rick LeFaivre, they did that and, unfortunately, they also had budget cuts and so even though they were transferring people out of ATG they weren’t having the budget to hire new people and transfer people into ATG. The result of that was that ATG became more and more research-oriented. It was the people who didn’t ever want to transfer to product development who were still in research and it was kind of like a university kind of department. We’re doing things for the long term. They should turn into papers. They get published. They should inspire people rather than become product templates and Apple could ill afford that in the 90s because, as you said, the money was no longer rolling in and they were running out of ideas and this group wasn’t giving them useful ideas anymore. All the useful ideas had already been transferred and they couldn’t come up with new ones because they couldn’t hire new people. So that was bad and in 1997, beginning of ’97, when Steve Jobs was half back, Gil Amelio was still running the place but kind of doing what Steve had recommended he do, Avie Tevanian, who was running software at that time, came to me and said, “We’re shutting down ATG. People say, oh no, we’re going to lose a lot of great ideas. Since you started it, would you go over there and find out what the great ideas are that are still there and which ones we should save and which ones we should kill?” So I was sent back there to kill the organization I had founded, but I’m glad he did it. It was the right thing to do. When I got there, I couldn’t believe what I was seeing. They were all really cool, but they were irrelevant. There was one that was Microsoft Surface, so it was 1997–2012, so 15 years. It was 15 years ahead of its time. There were no 18 months to 5 years or even 5, 10 years. It’s like, oh god, <*laughs*> this is way future <*laughs*> and their application of it was a conference room with people sitting around a big glass conference table with screens underneath it and then they would have a touchscreen of some sort on top and they would operate and have a conference. It would facilitate a conference. I said, okay.

Kossow: Sounds like what Engelbart did.

Tesler: We could sell— Yeah, he did something similar— we could sell 10 of these at a time. This is a company that comes out with products to sell in the hundreds of thousands and the millions, not in the tens, wrong company and so I killed that and I killed a few other things, spun out a couple of things. One of the things I spun out was to— myself it ended up because the team that was doing it convinced me that I should come and join their company and run the company that was something called Cocoa that is not what Cocoa is today at Apple. We didn’t get the trademark, but we got <*laughs*> the technology and it became Stagecast Creator, which I did for a few years from ’97 to 2001.

Kossow: Larry Yaeger’s handwriting recognition was going on then too.

Tesler: Yeah. So we kept the little handwriting recognition project that could be useful to the Newton because Newton was still alive at that time. We kept something that was called QuickTime for Java, which made it so that Java, which was very new at that time, Java programs could use QuickTime and we convinced them that was important. Actually, I heard what happened was that they actually canceled it for one day <*laughs*> and then reconsidered and talked the guy into coming back and giving up his package. And then— oh, Sherlock— It was the search technology that evolved into Sherlock that was like what’s now Spotlight. It may not resemble in any way the technology we were doing at the time, but we said that was an important thing, being able to search your hard drive.

Kossow: Maybe we should go back a little bit to the early 80s and to Newton.

Tesler: Well mid-80s really. I’m not sure exactly. It was, I think..

Kossow: Well Mike Culbert started in ’88 so …

Tesler: And that was pretty much near the beginning, so when I joined it was three years old, so it started in ’87. And Steve Sakoman was— announced that he was leaving Apple. He didn’t announce it. He didn’t get very far. <*laughs*> He got to his boss, Jean-Louis Gassée. He was managing hardware engineering or engineering, I don’t remember— I think hardware— and hadn’t been for all that long really and he was approached by Jerry Kaplan— I never got the story totally straight, did they dream it up together, did Jerry approach him, whatever— they came up with this idea they were going to do a pen-based computer and kind of Dynabook-like thing with a pen and he was leaving to do that. I think you should talk to other people about the real story. It’s a little more convoluted than that, but the summary of it was that he was going to go, Gassée maybe even was going to go along and run the company. I’m not sure how that all— whether that was a rumor or a fact. But Gassée talked him into staying and the way he talked him into staying was by convincing John Sculley that Apple should do a pen-based computer and they’d start from scratch— Jerry Kaplan hadn’t gotten anywhere, they were just getting organized— and they would do their own and— Knowledge Navigator kind of thinking, this is the future, we’ll do this.

Kossow: And Porat was proposing CRISP? [Editor's note: Porat was not related to CRISP. Porat proposed "Paradigm" also called "Crystal," which evolved into General Magic.]

Tesler: No, that was later.

Kossow: That was later?

Tesler: Yeah. That didn't come until 1990. So they started the Newton project and Gassée and Sakoman made some promises to the employees that we would later regret <*laughs*> in terms of company commitment to it, the rewards that they would get, but mainly, the bonuses or whatever they'd get for succeeding to motivate them to make it a real product, but the main thing they promised them was that there would be no marketing people, it would be all engineers and Gassée and they would build the engineers dream machine with no censorship from marketing people or finance or anybody else. This would be a pure engineering play with manufacturing, engineering people as well. And that's it and boy when I got there <*laughs*> I had a lot of angry people because, "we said we could do it anyway we want; it doesn't matter that it's going to cost $7,000 and be seven pounds and be larger than a notebook; we don't want to make any compromises; Gassée said, 'no compromises' and so it's going to be the best of everything." And, well ...

Kossow: So were there any compelling features to it at that point?

Tesler: It had an idea for networking that would be great if it worked, but they were still deciding when I got there in 1990 what technology to use, some kind of spread spectrum. It was unclear what they were going to do. Should we make it a standard, should it just be for Apple only? Should we adopt FireWire or not because that's being developed for the Mac, but it's not out yet. It may be done too late. But we don't want to not use SCSI. There were just lots of open issues when I got involved. Handwriting recognition, one guy was working on it. Cursive handwriting, when you asked him for a demo, he kept saying, "I think I'll be done in a week or two," and that happened every week. It didn't really have any way to sync with your desktop machine. It had a very powerful processor that didn't exist yet that they had discovered at AT&T based on, I guess, what is called the SPARC. Not the SPARC <*laughs*> that's the Sun…

Kossow: The CRISP?

Tesler: CRISP, yeah, it's the CRISP, AT&T CRISP and some evolution of that kind of stack-based architecture into a low-power stack-based architecture, which was called the Hobbit, AT&T Hobbit, which was for Apple and it was going to be an exclusive to Apple for a certain amount of time and the Newton team considered that a plus that it would be exclusive to Apple. So I got there and all these decisions had been made and I really disagreed with almost every decision. However, I was taken by the idea of a pen-based machine and the fact that we could build something portable and carry it around and worried

that Jerry Kaplan’s company might get there first. Soon after that, Microsoft announced Pen Windows and it started becoming a competitive thing: Even if we don’t know how to do it, what if one of them figures out how to do it, we’d better have an answer. So we can have a machine that’s 90% figured out and once they figure out really what the right thing to do is, we’ll finish the last 10% and we’ll go in the market, too, but we better be started. There were a lot of people arguing we should do it with a Mac architecture, but a lot of us engineers who were a little bit more future-oriented were always saying, oh no, we could do a whole new operating system a lot better and be better fit than a Mac operating system. And I have to say that it was the right thing to do for iOS to do what the Newton did and be a different operating system for the iPhone, but it was the wrong thing to do, back then, to do a new operating system for the Newton. We could’ve used a Mac operating system. It would’ve probably been two or three major versions behind whatever we were using on the Mac at that time, but we could’ve focused our energies on developing handwriting and networking and application software kit and everything else, better UI. We wasted a lot of time developing our own operating system. In the end, it turned out to be a very good one, not when the Newton first shipped, but a few years after it first shipped. The handwriting got better, the, especially the printed handwriting. The operating system got better. The hardware got better, but the Newton was just a cursed name and you couldn’t have a product called the Apple Newton, doesn’t matter what it did. It could go to— take you to Mars and back and no one would be interested.

Kossow: So there’s also a bunch of early work on wireless networking that came out of the Newton Group?

Tesler: Yes. The wireless network. Originally, it was going to be— sit around the table— instead of having a glass table looking through it, it was something a little more reasonable, which was: everybody would have their Newton— Now what if not everybody has a Newton? Don’t worry about that. Everybody will have their Newton— and they will communicate. People would get their turn to drive the big screen. There will be a projector. Everyone will see what’s on their Newton up on the big screen. They’ll distribute documents among themselves. They can shoot private messages to other people in the room. Pretty good, pretty interesting ideas. And there was one, I think one, hardware guy that was kind of the architect, Jim, help me, maybe it’ll come to me. They usually do. His name was Bill Stevens. He only worked for me for a few weeks because one of the things we decided early was that— he was pretty honest about how long it was going to take to finish this and it was going to be too long no matter what schedule we considered reasonable for the Newton and we needed to get the cost down and initially this would cost a lot even if long term it would be super cheap, but also you can’t have a network without a standard. We already knew that at that point and so we decided let’s move this into ATG. We’ll swap places. So Bill went to— into ATG, started working on it over there, got on a standards committee— or formed a standards committee— convinced people there needed to be one and got David Nagel to go make trips to Washington to talk to whatever was necessary, FCC, other standards bodies, other companies that we wanted as partners, and developed the Wi-Fi standard, which has evolved into what we have today. And recently I ran into one of the guys that I think was the guy who actually started the committee, the main guy, and he acknowledged that the major source of ideas at the very beginning was Bill, who had worked on the Newton at Apple. But a lot of other people recognized the need and they

started the standards body, but he arrived with, hey, we’ve already thought about this and here’s some ideas.

Kossow: So then after Newton there were a couple of communications things that you worked on, did you work on eWorld?

Tesler: Yeah, after the Newton. I left the Newton when it reached beta because it was really alpha— code complete— and the new head of the division declared that it was “beta”. It was nothing we could send out to users. It was too flaky, but finally it got reliable and we called it “beta” and we could put it out to users. He called that “final” <*laughs*> and so they shipped the beta. And so on top of the inaccurate handwriting recognition and price and other things, it was unreliable. So that contributed to it bombing and I sort of relived my Lisa situation again. I left a few months before it shipped and it wasn’t good. I don’t think I would have stopped the Lisa from shipping except for cost reasons if anybody would let *me* do that, but we’d already decided to let it be that price, agreed to that. Well, I couldn’t go back to ATG because David was running it, but I was happy to report to David. They just had to figure out some title for me and I would say, Oh, it’s okay. I’ll go do the next thing <*laughs*> after the Newton and I don’t need to run ATG. Before I had already had experience growing a group, downsizing it and leaving, or handing it over to somebody else and leaving it and starting another group, so it didn’t bother me. I wasn’t on a career ladder. I was on more of a career seesaw, didn’t bother me. Come up with a new thing, grow it big, give it to somebody else, start over with a new thing. But so here I was starting over with a new thing again, had to figure out what to do, and I decided scientific visualization could be an interesting thing to work on. It was something that I got into a little bit on my sabbatical in ’87, I guess, ’86-’87, where I went on a sabbatical with my wife who’s a scientist and worked with her on some visualization of geological images. But at this point, scientific visualization had grown to a lot bigger than just looking at image data. It was also looking at data coming from 3D models of things, and sort of Tufte kinds of visualizations, and so it was a hot field and there was a lot of work going in SIGGRAPH, NCSA and so on. So given that we had a big scientific and engineering market of about a billion dollars a year at that time and we had a big academic market of about a billion dollars a year and that was out of— it peaked at seven billion before it started going down— and it was at that point starting to go down, but those markets were holding up and I thought we should do something for the scientific and engineering market, otherwise we’ll just lose it, the way we lost other markets we’ve tried. So I did that. At that point, I guess Nagel recommended that they make me chief scientist of the company and I did that for a few years. Mostly it was a titular kind of honorary role, but I had a couple of responsibilities. All the Apple Fellows reported to me and all, what are called the distinguished, the DEST, the distinguished engineers, scientists, and technologists, I put that group together for the level of engineering just below fellow and you didn’t have to do an earth-shaking thing like Elkay [ph?] did to get to be a Fellow, if you get to be a DEST, but you had to be one of our top something percent, small percent of engineers, scientists, and technologists. And so we started that program. And so we had periodic meetings where we would talk about technical issues the company had and what we should do about them and made recommendations to the management. The most notable thing we recommend, two things we recommended, was that they cancel the operating system that was the successor to the Mac operating system that they’d been working on for a few years because it looked to us like one of those projects from hell that never end and always look like you can see the

light at the end of the tunnel, but it’s just another tunnel. I can’t even remember what the name of that one was.

Kossow: Copland?

Tesler: Yeah, Copland was the name at that time and the other thing was OpenDoc. We had another meeting where we decided OpenDoc was a bad thing. Actually, I take it back. We never got to that meeting. <*laughs*> I was going to call a meeting to discuss it and Nagel had me go around to get a sanity check on this recommendation about the OS. He wanted me to personally go around and see if it was really true that it was hopeless— the operating system— and I said, “I also want to see about OpenDoc. A lot of people are complaining about that too,” and so I did and all the OpenDoc people said we should cancel the operating system and all the operating system people said we should cancel OpenDoc. So I came back and I said, “I don’t know. I’ve been an engineer for a long time and I think we should probably cancel both,” and they lasted a while longer. They didn’t get canceled, but they died a slow death. We probably should have canceled them. But he— at one point he—asked me what I wanted to do next after whatever I was doing. First, I went to eWorld. Oh, I remember what happened. I actually resigned <*laughs*> and I said I been here long enough. I’m going to resign and do something else. The Internet is happening and I want to get into the Internet business. This was probably ’95 or something and Peter Friedman came to see me and said, “Ah, I’ve got this group called Apple Online Services, eWorld.” He explained the whole thing, talked me into going there and starting a research group inside there and mainly they were dependent on AOL technology to build their community and I said, “Why don’t you use the Internet?” and he said, “Because we don’t have anybody that knows the Internet.” He said, “Why don’t you come over.” I said, <*laughs*> “I don’t know about the Internet either at that level of detail just at the higher level. I’m a user of the Internet but not a developer of it. Done a little networking software but hardly any,” and he said, “Well, this is an application level group. You don’t have to build the Internet, just build on top of it.” Okay, so I went there and I spent maybe nine months at eWorld trying to build an AOL— or eWorld actually— clone on the TCP/IP protocol instead of on AOL technology. And we just started getting it working.— It was in a group in Boulder, Colorado. It was little far away to manage them very well.— And then all of eWorld got canceled and this project got flushed with everything else because it was the smallest part of it. They just wanted to get rid of it all, save money. So I resigned again <*laughs*> and David said, “If you stay, I’ll put you in charge of anything you want,” and I said, “I’d want to do Internet stuff.” He said, “No. I won’t put you in charge of that.” <*laughs*> He said, “Anything reasonable that you want.” <*laughs*> So I said, “Okay. How about just the application layer of networking?” And we finally concocted a group that we called the AppleNet Division and it was just random things. It was the AppleShare file server— which wasn’t an Internet thing really, had its connectivity, but it was a LAN product— Cyberdog, which was a browser built on top of OpenDoc, and this Cocoa project, which was going to be something like developing network applications for kids or something like that, some justification for being in the AppleNet group. So it was kind of a hodgepodge of projects and...

Kossow: But this was in ATG?

Tesler: No, this was not ATG. At that time, Nagel was already running all of engineering at that point and I was going to say all of software engineering, but I think at that point really all of engineering. Actually, he left shortly after and went to AT&T, but before leaving, he set me up to run this AppleNet group. And I had a great time. There were two people reporting to me for the covered hardware and software. There was one other person reporting to me just to cover this Cocoa project, which really didn’t fit. It was just one of my favorite projects and it was Lynn Welge doing software, one of the greatest managers I have ever met and— she actually worked part time as she was managing a, I don’t know, 80-person group or something and flawlessly— and Buzz Dean,who was doing the servers. And they were both sort of dream subordinate managers, <*laughs*> have a meeting every week, go through your to-do list, what got done before, what are you going to do next week, come back next week— they did it all on schedule— What’s next? It was like problem-free management and when I offered them ideas, which was once in a while, they actually considered them and if they were good ideas, they’d use them, and if they weren’t, they wouldn’t and didn’t ever do it just because I said so and just wonderful. I’d never had managers below me that were that good and it was a delightful time, but it didn’t last very long. When Steve came back— which I’m glad he did, <*laughs*> I was glad he did at the time and I’m still glad he did— everything got reorganized, so there weren’t business units anymore. It was actually one of the best performing business units. I mean, when Avie came to me and explained the new structure, I said, “We’re going to make money this year. We’re going to make a profit.” This is the only division that’s going to make a profit because I had these two great managers and they got out profitable products. He said, “Nah, it doesn’t matter. We want to reorg and be functional. And in the long run, that turned out to be the right thing to do. I was puzzled at first. So that was it.

Kossow: And so at that point you were working for Avie or..

Tesler: Yeah, from the time we acquired NeXT end of ’96, I guess, beginning of ’97, until I left August 1, ’97, I reported to Avie Tevanian, who was running software. No, not the whole time, I take it back. <*laughs*> At the beginning I reported to, I think it was, I had backward order, I think I reported to Ellen..

Kossow: Ellen Hancock?

Tesler: Ellen Hancock. Then, she said, “Things are changing. Temporarily you’re going to report to Gil Amelio.” I knew it was temporary, so I reported to him for a while. He was CEO and he would have me come in and run ideas by me on the whiteboard and get my input and then he started getting input from me about who should we acquire— Be or NeXT? And so I was peripherally involved in that, but I had no idea ever what the status was. I mean, he just wanted my opinion of very narrow things, didn’t want to give away too much. Until the very en— then, he wanted my final opinion and then he told me what’s really going on. And NeXT was my recommendation. But I worked for Ellen, then I worked for him, and then, I don’t think I worked for him at the very end.

Kossow: Were you still chief scientist at that point?

Tesler: Somewhere in there I think I reported to Avie for a few days, but a lot of turmoil was going on. I was still called chief scientist. I was still VP, but in the last few months— oh, that was it, when he came back to me sometime around, I think, February he said, "You got six months to close down ATG," and it didn't take that long to decide what to do, but it was right up to the very end of July when we closed it down and I left and— yep.

Kossow: So had you had any ideas on what you were going to do after Apple at that point or?

Tesler: Well, I had already tried to resign twice to go into Internet and so that's what I would've done, but sometime around June or so, the Cocoa people— I'll call [them] Stagecast people because that's what it became, that name came later— were friends of mine. I loved their project and they came to me and they said, "Is this project going to survive the cut?" and I said, "No. The most optimistic scenario ever, you wouldn't make enough money for this to be a blip in Apple's P&L. It's just too tiny. We got to get rid of it." It was educational programming for 7 to 10 year olds and— visual programming. I thought it was the coolest thing I ever saw, but I admitted it wasn't big enough for Apple but might be big enough for a spin-out company and they persuaded me to run the company and I said, "Well, if I can talk people into giving us capital, then I maybe I can run a company <*laughs*>." So I went out there and started raising capital. It was very hard, but I did, got a bunch of angels and then a VC and I thought, oh okay, now I can do anything and we got the product working. We got a successful introduction at a conference and then it was all downhill from there and the company failed, still actually in business. The product is still for sale, but there has been nothing...

Kossow: Just nothing, no traction?

Tesler: It never grew. It was a small market and never got any bigger and the software hasn't really been improved, just updated for compatibility with new operating systems. I am still fond of it, but I have no connection with the company any more.

Kossow: So then after that you were at Amazon and then Yahoo?

Tesler: Well, yeah, I was at Stagecast for four years, wound it down, tried to get it into good shape for just becoming a slightly profitable company that could pay off its debts essentially and give a couple of people a living and I started job hunting and I talked to a few companies. The one that was most aggressive about wanting me was Amazon, but they were in Seattle. The other company that was interesting to me and a lot easier on my life <*laughs*> because it was in Silicon Valley was Google. So I interviewed at Google, but I was warned by Wayne Rosing who had gone there and was running engineering, he wanted to have me report to him and start an engineering group so he wouldn't have all engineers report directly to him, which was the case at that point. He tried to convince and I tried to convince Google management to hire me, but we were unsuccessful. At that time, every employee had

to be approved by certain people and somebody vetoed me. But at that point, I already had an offer from Amazon that was really interesting and I'd been figuring out how I would deal with a job <*laughs*> in Seattle if the Google thing fell through. So I executed my plan and went to Amazon. I had to reschedule my interviews because first interview was on 9/11/2001, so we had to change that. But I ended up going there first of December, I guess, and staying there for three and a half years. I started out managing a couple of engineering groups and did a little bit of that while I was there, a lot like Apple. People would, "hey, can you manage this engineering group for a while until we get a permanent person in" or whatever, just having a lot of experience you get stuff like that. And so I managed several different things. But my first love was the user experience aspect of it and that's what I wanted to focus on. And starting about April of 2002 until I left April of 2005, I was running sometimes other things also, but mainly running what came to be called the Shopping Experience Group. Then, I went to Yahoo just because I missed Silicon Valley. I was starting to notice when I came to visit that I knew fewer and fewer people and I thought, this is not good. <*laughs*> My network of people in Silicon Valley is really important and it's falling apart. In fact, it took me several years to rebuild when I came back. But mainly, my wife was here, my mother was very ill and aged and here, and I just wanted to come back here. So I did and I interviewed at Yahoo and one other place. The other place and I weren't a fit mutually and Yahoo seemed like a very good fit. I'd also tried to stay at Amazon by going to A9 or Kindle or something, which were both pretty new at that time, but that wasn't easy to make happen and I had an offer from Yahoo I didn't want to delay answering to pursue the possibility that I might be able to stay at Amazon. So I went to Yahoo and there I was again running user experience. In this case, not just usability but also the actual designers. At Amazon, the designers didn't report to me, but at Yahoo they did. And I did that for three and a half years. I had a second role there which was a fellow basically, "research fellow", they called it at first, then just a fellow, "Yahoo fellow" and that was initially just a title that I had, but in the last year I was there I actually started working about a day a week on more research stuff having to do with ways to manage advertising campaigns, just to put it in a nutshell, and some other things that became patent applications. But the main thing I did at Yahoo was to decentralize what was a centralized group when I got there for various reasons and at first it seemed like a good idea, but what I heard was later when they hired somebody else to run this group, this, user experience design group, Tim Parsey, they re-centralized it. A lot of companies go through constant cycles of centralization and decentralization. I think Apple has proved over the years that it might be better not to do that so often. They did a lot better when they just went to functional and stuck with it.

Kossow: Can you see any of the Apple influences in the Valley now that people have spun out companies and ...

Tesler: Yeah. There's a lot of influences. Yeah. A lot of it is the people who used to work at Apple both in the early days of Apple, the era after Jobs returned and the short time since he's left us. People left Apple during all those times and seeded many, many companies, I couldn't even start to list them, either as the founders or as key people in those companies, but it's really true of any large company in the Valley. You could say that about almost any large company, but I see Apple a lot represented as alumni in a lot of places. But the other thing is just the influence of Apple's focus on design, on simplicity, on usability. For years, people would say, well, Microsoft and Intel are way bigger than Apple and Motorola,

so it doesn’t matter. It just proves it doesn’t matter. But I think people began to realize that whatever the reason it didn’t matter then, it matters now <*laughs*> and you can argue why, but I won’t take the time. And just companies are falling over themselves trying to find people who can transform their culture from an engineering culture or a finance-driven culture, or whatever it is now, to something that’s maybe design-driven, or maybe just more balanced, which is probably for most companies the right answer, a more balanced approach. Get away from a model before where designers were actually disrespected and limited in their career growth to one now where maybe idolized too much <*laughs*>, but certainly respected well, and given career opportunities that they couldn’t have before, going higher up in management. So I think that’s in many ways a reaction to Apple and other companies that have done similar things, Google for sure and a few others, but Apple more than any.

Kossow: Great. Well, thank you for your time. It’s been fun.

Tesler: Thank you.

END OF PART TWO

END OF INTERVIEW

