Oral History of Larry Tesler, part 1 of 3

Interviewed by:
Hansen Hsu
David C. Brock

Recorded November 22, 2016
Mountain View, CA

CHM Reference number: X8020.2017

Hsu: Right. So today is November 22. I am Hansen Hsu and this is David Brock, my co-interviewer and we are here with Larry Tesler, or Lawrence Tesler.

Tesler: Larry.

Hsu: <laughs> And who will be talking to us about his experience at Xerox PARC and about Smalltalk. So we're going to start maybe with right before that part of the story with your experiences at the Stanford Artificial Intelligence Laboratory. So how did you—how did you start there? You were working as a consultant? You had your consultancy job practice and you were contracted to work at SAIL at some point?

Tesler: Yeah, when I was in college at Stanford. I started a little consulting company. At first it was just me, but at the high point I had four employees, mostly students, and for some of them their first job. They were still in school, so was I. One of my clients after I graduated was Ken Colby up at the A.I. Lab, and he was a psychiatrist who had developed an interest in using computers as an adjunct in therapy, and he hired—he had hired some friends of mine as employees, and when they heard I was out of college and available for consulting they suggested he bring me in to complement the others. I had a strong interest in natural language technologies and one of my clients had been the head of the linguistics department at Stanford. So that piqued my interest even more. And so that was the part of what Colby and the A.I. Lab did that I got involved in—[it] was natural language mostly.

Hsu: So were you—did you have any overlap with—did you have interactions with SRI during that timeframe—you were in the A.I. Laboratory, SRI was sort of a separate group.

Tesler: Yeah, SRI was still part of Stanford in the—I think until after I left the A.I. Lab even. I had some interaction with the people in Doug Engelbart's group, the Augmentation Research group. And I'm trying to remember now—oh yeah, one of my clients, different from all of my other clients, didn't want me to write programs for him. He wanted me to help him immerse himself in the computer industry as a businessman, and so he would come ask me questions. He would meet with somebody in the field, not understand half of what they told him, then come to me to translate. One day he said, "I heard there was an amazing demo at a conference in San Francisco last week." He said, "Did you go?" I said, "No, I didn't go." And he said, "Well, I called and got permission to come over and bring you with me to get a private demo." And okay, whatever. So I went and saw not Engelbart himself but one of his employees give a—give a demonstration that was somewhat similar to the one in San Francisco. Anyway, I was very blown away by the power of it, and at the same time aghast at the usability of it. <laughs> And usability was something that I took an interest in from the very beginning when I got into computers, which was in 1960 at what was it, at the Bronx High School of Science and somehow weaseled time on a computer at Columbia University and started learning Fortran and immediately sort of just objected to [the] terrible

syntax of it. Why was this—everybody's going to have a computer someday, it was obvious to me, and it needs to be easier to do than this. So I got involved in that, and that—I then had periodic encounters with people in Doug's group, but nothing like what happened when I—when I moved to Xerox later.

Hsu: Was there a difference between the cultures of SAIL versus SRI?

Tesler: Good question. Yeah. Well, it's hard to compare SAIL and SRI, now that I think about it, in terms of SRI had lots and lots of buildings and many people and government contracts, and SAIL also had government contracts but just one part of one building and not a lot of—not a lot of people, but they had a few people who were good at getting grants, and that was a period of time when A.I. was positively regarded by people in the field. Later, when it disappointed people they had a more difficult time. But my first work for SRI was actually a few years before that, 1962 or something where I was a computer operator on the Burroughs 220 in the basement of Encina Hall, and there was a nightly run of nuclear fallout simulation for the Civil Defense—Office of Civil Defense, and I got intrigued by the printouts that were kind of graphical printouts on a text printer, and again got me thinking of what if we actually had graphics printers and interactive simulations on screens and that sort of—that sort of desire, and people often say that nobody could anticipate what computers would do in those days, but all my Stanford friends and I had it pretty much figured out a very high level. Once in a while something would surprise us, but mostly on the positive side, like whoa, that's what we were thinking, but even better.

Brock: Could you describe a little bit of that vision that you and your computer minded friends had in this era about what you thought the computer had to become?

Tesler: Well, mainly it had to become something that everybody had access to, and for a while there was John McCarthy's time-sharing concept that people thought was maybe the—would be the first practical way to do that. But computers were too slow to share. It was already very little—you already got very little compute time on a machine. To have to actually share it with other people and with the task switching overhead it was just not practical to use time-sharing. A lot better solution, once the computers got down to a certain price range would be that everybody had their own computer, and some of us negotiated borrowing our minicomputers and bringing them home and working late at home. But they weren't really things you could carry around the way that we're used to now, and but we knew that was coming, and computer graphics. I did a computer graphics project in 1962. Some older students, Larry Breed and Earl Boebert at Stanford had taken the Stanford football halftime card stunts and created—software that would allow an art student to define frames of an animation and action. And when they finished they found that they didn't make it easy enough for the art students to do, they had to do it themselves, and that wasn't what they bargained for. They were—they were leaving to go to graduate school anyway and they had to find somebody to take it over so what better than a—than a freshman.

<laughter>

Tesler: And that was the end of my freshman year so they got me to stay around in the summer and do a little work on—for systems software there, and but also to take over the management of the card stunts. And they said, "If you—" Larry told me—Larry Breed—if let's see—oh yeah, "If you don't make this easy for them to use you're going to have to do it yourself, and you won't like it." And so I did—I did the first couple for the upcoming season, there was no time to do anything else, but I modified the software so that it would be easier for an art student to use, and developed a usability testing sort of protocol of my own because I didn't know such things existed, they did at IBM and places that had affiliations in psychology labs, that sort of thing. And that got me interested in making application software that was easier to use, and when I started consulting in '63—I did a little consulting before that even, but in '63 I started a—I incorporated a—started a company to do stuff more officially. I needed a corporation sort of to hide behind because I was only 17 years old or something, and I didn't want people to assume that I didn't know anything because I was 17 or 18 years old. Anyway, other visions that we had. Well, robotics, that was one that took a long time, to do some of the more interesting applications of robotics, but just basic industrial robotics was already getting started at Stanford and other places. So that clearly was going to come and we knew that would be hard, to have unbound robots that could move around and do things and act intelligent. In fact, a lot of the things that we saw coming were things that could be done in the near term, like networking, for example. And other things like artificial intelligence were probably mostly going to be things for the future, and that is how—what came to pass, and one reason I and other people left the field of A.I. was because we realized how long it was going to take; we were going to be retiring by the time that it did anything useful like that we—in our dreams. But we had the training of A.I. and that contributed I think to the skills that we were able to provide in computer graphics and computer games, and all the other things that kind of started around then in the sixties, and took a while to pervade our culture.

Brock: One other quick, follow up question if I may. The fellow that you worked with at SAIL interested in the connection of computers and therapy, was that—so sort of could you talk more about that? Was that to get the computer to understand natural language so that it could participate in talk therapy? Or—I'm just intrigued by that.

Tesler: Well, that's the right thing to think of first. But because he was not a computer technologist in any way, but more of an application visionary like Engelbart, like Bob Taylor, he had the visions—Ted Nelson [also]—had the visions, but not the technical skill and know how, just got into the field a little late. Most people need to learn programming when they're learning—when they would be learning music or math, very young. And the—what was your question again?

Brock: Just what his idea was with the connection between therapy—

Tesler: Oh, Colby. So when Colby started out he thought that—he imagined that some of the things that were being done by therapists could be automated because they didn't really require knowing anything, and so he was really not looking for intelligence at that point, that could come later, he wanted to see

what you could do without intelligence, but just sort of pretend to be intelligent. And he was drawn to the Turing test as something—Turing's idea was the way to define intelligence is by seeing if you could have a computer fake it, fake being male or female or human or computer or whatever, and so that was clearly a pretty limited amount of intelligence, but if it worked then that could be—that could be useful. And so one thing he did was recognize—well, having written a textbook on psychotherapy before, Rogerian therapy, he knew the kind of the script that the doctor went through in order to screen the patients and put them into gross categories, and then get them ready to go to a therapist who could do the things that were not so mechanical, things that were more creative. And so he implemented, or his—my friends there implemented a program called PARRY that was—that simulated a paranoid patient. And paranoid patients don't always respond to questions in a relevant way, which he was—he used that to cover the fact that the computer wasn't really understanding anything people were talking to it about, just but there were trigger words that the paranoid person reacts to. So it wasn't that hard, you just ask a question, see if any of the trigger words were there. If it is, say something totally bizarre and you'll qualify. He also did something he called the Mad Doctor which was somebody to simulate a psychotherapist and so you could log into the Mad Doctor and then say, "I'm really frightened of heights and I don't know what to do. In my job I have to be high up." And again, they just search for certain keywords and based on the keywords being there or not it will say, "Oh, I can help you with that," or just ask a follow up question, repeating some of your words that it doesn't know what they meant at all. And just say, is it—"Is this a daily or hourly kind of thing that you x?" Anyway, it was—Joe Weizenbaum from MIT heard about it, and came and visited him and got inspired to do what became ELIZA, and they had an interesting relationship over the following years, but that's a long story.

Hsu: So you mentioned that from very early on you were interested in usability, from even in high school. That's very—it seems to have set you apart from the other computer people in college, the people that had developed that card stunt program at Stanford. Was there anything unique about you or your experience that made you more attuned to ease of use?

Tesler: Well, I think it was a combination of some things about my own inclinations, but it was Stanford also because those very students—well, Larry Breed at least was my mentor when I was a freshman and sophomore, and he very strongly backed my doing stuff like that. He didn't have a lot of ideas about how to make computers easier to use. In fact, he did the first implementation of APL, and definitely not something that your average person is going to be—have an easier time. Maybe your average mathematician that specializes in matrices or something. But within his scope of projects that he did he always was looking for a way to make things easier for whoever was providing inputs to this program, and so that encouraged me quite a bit. And I kind of got a reputation around the campus for being good at that, and so the kinds of contracts I got were ones where somebody was interested in running the same program over and over again with different inputs, and they were worried about usability issues. And hey, there's this Larry Tesler guy, you should see if he's available to help you. And so I got a lot of my business generated that way, which encouraged me more. And so part of it was my interaction with other people and positive feedback when I made things easier, and just also just made me feel good to see that people could do stuff easily that they couldn't—couldn't do before.

Hsu: Earlier you mentioned Ted Nelson and around—there was a period where you were teaching at Free University. You were instrumental in getting Jim Warren interested in computers.[1] Could you talk a little bit more about your connection to what was going on in the counterculture and what was going on at the computer laboratories at SAIL and at SRI, and how—whether those values motivated you and influenced you in pushing for better access to computers or more usability for computers?

Tesler: A lot of material in there, but no, that's okay. Just trying to decide where to start. What's the thing you're mostly wanting to get at here? Is it the counterculture?

Hsu: The counterculture. Did you follow Stewart Brand—and since—?

Tesler: Okay, so there was this organization called the [Midpeninsula] Free University. And I got married very young and got divorced very young, and wanted to make new connections with people after my brief marriage. And somebody who I knew told me about the Free University, or gave me a catalog and I went through it. "This is really interesting." Just something about it. It was just rebellious enough to say, "Well, your Stanford education is not good enough, you should take a—take a course in Palo Alto like a community college or adult education sort of course—or teach one." The thing that was more unusual maybe at that time was getting people to teach courses[2] [for free]. And all sorts of subjects, from ones that were very academic to ones that were really more just like having a party basically and calling it a class. It was an unaccredited—and no intention to be an accredited—university. But I got involved in it. And I had a little business experience from running my consultancy, and they needed a treasurer, and that was one of several times in my life that I became treasurer of a nonprofit I was involved in.

Tesler: The—I think, well, John Markoff has written very—and other people have written very good histories really, and with biographical touches throughout, of people who were involved, and the—it's probable, I would say likely, that in some manner the personal computer revolution, which is now a historical era of our field, was accelerated by the counterculture movement because people—all of a sudden there were a lot of people who wanted computers to be easier to use because the prices had—were coming down to those kinds of levels, and time-sharing was around for people who couldn't get access to a minicomputer, say on a regular basis. So that was all happening, the technological capability was happening.

Tesler: As I said a lot of people had visions of what it would be like to have your own computer or a lot of time on a computer, and the counterculture helped in various ways. One was after a protest, or before a protest even, they would—the students who were rebelling would crash into an office building, run flyers off on the offset printer in that building. In the meantime, in that process learning about printing

[1] Larry Tesler adds: I used to think so. But, according to Jim, he was employed as a computer graphics programmer before we met. I've confused him with someone else.
[2] Larry Tesler adds: The courses were free.

technology and things that software could make easier to do, and kind of doing both in their life at once, being a member of a—of some kind of movement, political movement and being a contributor to the early days of the computer revolution. They were kind of happening at the same time, and people naturally noticed the interaction points and leveraged them.

Tesler: And so we were talking about Jim Warren. We [Jim and I] were making catalogs and newsletters for the Free University, and that whole time talking about the future of computing as our background subject. We'd talk some about the course topics—Esalen massage, political heroes, [etc.,] but also we would talk about what was coming. He had been a math teacher, high school math teacher and chairman of the Math Department of a college—and was at the time a graphics programmer for the Stanford Medical Center. I had been a math major, and so naturally computing was of [mutual] interest.

Tesler: Another volunteer I worked with had played with computers a bit but never with graphic screens, didn't even know that existed, and one day I said, "You know, someday what we're doing will be done on a computer, we're doing cut and paste." And he said, "How could you do that on a computer? That's just—" To him a computer was something with a typewriter, and so I described Spacewar! and his eyes lit up. I said, "Come on, I'll show you, I'll show you Spacewar!." I think I took him to the A.I. Lab and he got involved there, started working at the A.I. lab as a volunteer at first, I think, and then got deeper and deeper into it, and just created a whole new career for himself in computing.

Hsu: So you were—you were already thinking in terms of copy paste, even at that point.

Tesler: 1968. It was around the time of me seeing Engelbart's demo, and I—someday I should try to reconstruct which happened first, the course catalog session where I started talking about cut and paste and it gradually became an obsession with me—versus when I first saw Engelbart's stuff, but they were around the same time and I—for example, it could be that I saw Engelbart's stuff and went, yeah, except that the way they move and copy text is just very ponderous and error prone, and I did a lot of calculations to show that you could edit with half the interactions if you used a different method, which was to do what we used to call prefix commands—no, to do postfix commands, I'm sorry, rather than prefix commands. In other words, do you specify the verb first or do you specify the noun first? If you want to delete the third sentence of the paragraph do you start by saying delete and then have a way to refer to the third sentence of the paragraph, or do you first select the third sentence of the paragraph and say delete? And both styles existed, and I realized that there were pros and cons and but that the one that—the winner was going to be the one that was easier to use and had good marketing potential.

Tesler: And so that was part of cut and paste, cut and paste is—[in] the whole printing industry, everybody knows cut and paste, but then what about everybody else? They don't really know cut and paste, but it would be not very hard to learn. It's, if in some industries it's a common metaphor, maybe it's one that is just—is pretty easy. They're not technical people who do cut and paste in the publishing industry. And so I was torn between making it too hard for non-printing industry or non-publishing industry

people, and making it just too hard for everybody, which was what everybody was—other people were—doing.

Tesler: So I struggled with that and it wasn't until I was able to run usability studies at Apple—no, not Apple—at PARC, in connection with a project I was doing for a subsidiary of Xerox called Ginn and Company, that's when I was able to validate that not only was it easy and very very well-liked by people in that industry—they went, oh, I can cut and paste. That's great. They were sold before they even saw it, and but for other people, even people who hadn't really heard or understood what that term meant it was just instantly recognizable.

Tesler: I like to say that in those days computer programmers were always implementing move and copy and delete when they did their text editors. Insert, append, replace, these were the kind of commands you had, and do you want to do something before the selection or after the selection? Do you want to tell a computer what you want done first before you tell it what to do it to? Or these were—these had no answer. People knew these were questions, research questions, but they had no answers, but once I was able to just run a few simple studies it came out very strongly that it was way better to first select and then do the command. For one thing if you made a mistake in your selection, which was very easy to do with the mouse, especially in those days, just do it again. You don't have to tell it, I changed my mind, I made a mistake. Undo even. You just do it again. Now a different thing is selected, miss that, do it again, double click and drag, multiple words. So—and then once you've got the selection now you get to do a command, and if you do the command, if you tell it the command correctly you're fine, if you make a mistake we have an *Undo* command for you that will undo the last thing you did. Bingo. Graphical user interface. That's it.

Tesler: And so in terms of text editing, because things were slightly different for graphic editing and database editing and so on. Some things didn't carry over, but even in those databases and graphics, text was part of that. So text editing is a pretty fundamental use of a computer, it enables other things to be done that you wouldn't think of as text editing as an application area. And so I just knew that it was critical to get that right, and whoever got that right, it would take off and be adopted by everybody, and that was it. So I got good allies at PARC, and we got it working and addressed all the boundary conditions and edge cases and made something that everybody decided was it. To the point where in those days most people hadn't heard of cut and paste, but it was initially a way to do move, copy, delete, and replace… Insert and replace.

Tesler: But nowadays it's turned around. Nowadays people think in terms of cut and paste as being the primitive operations. If you say something like, what commands do you move and copy with? "Huh? What do you mean? I don't know what you're saying." In those days, everybody knew what you were saying, but they may not have heard of cut and paste. Now people say, the way you want to do that is by cut and paste, and that explains it to everybody. Oh, yeah, of course I know how to do cut and paste. And if you make an error and say, oops, that was a copy paste error—but everybody knows what you mean. It's

become part of the language and it's reversed from where it was in the sixties in terms of which is the primitive and which is the defined in terms of that primitive.

Hsu: That's fascinating.

Brock: Just a quick question that I've had for a while. At this time period when you're getting obsessed with the idea of cut and paste it seems to me that there is something going on in the computing—in computing circles, communities, of looking at text not from the vantage point just of its information value but text as a kind of an image, that a lot of these concerns and operations and the approach of working with text has to do as much about its appearance as form on a surface as it does its representational value or its informational value. Is there something to that, or maybe you could give me a better way for thinking about that.

Tesler: Yeah, text as—text as a kind of graphic art as opposed to text as a neutral carrier of information, like a telegraph. And a lot of us who got into that field were very entranced by typefaces, and Ted Nelson very much so talks about that all the time. Alan Kay and Steve Jobs. It's just a fascination with all these different ways you can write the letter F or something. And how that can be used in a decorative way as well as information carrying. I mean the choice of fonts carries information, this is a very formal document, this is a very technical document, and this paragraph is a quotation that somebody else said. And it—typography helps the writer to communicate things to the reader that are not said explicitly. So there was that fascination but computers were too small to carry around—if you said something like that where I'm going to—I'm going to have fonts on my—in my computer, people would say there's no room. There's no room for more than—even one font is kind of a lot to keep around. But everybody really knew that things were accelerating in terms of the speed of the computers and the memory size, the costs going down. I mean Moore's Law was recognized in the mid-sixties and it was unavoidable. It was if you had an idea it was very much the case that it was maybe a little early to have the idea because the technology couldn't quite do it yet, but by the time you took a breath it—your competitor went into it at just the right time and they benefited more from it.

Brock: Thank you.

Hsu: You mentioned your interest in printing and publishing even at that point. Where did that come from?

Tesler: That came from love of books when I was a kid, but also from the paste-up I was doing for the Free University. And the Free—at the Free U we purchased an offset press and started a little print shop in downtown Palo Alto helping to pay for the costs of our printing, first of all by doing our own printing, and then doing other people's printing also when it was available. So a couple of the guys learned how to operate an offset press, and I was one of the people who was generating the demand. Why can't we do

this special thing with the printer—I don't remember any of them anymore, but it was—we had to do the design for the covers and the selection of typefaces and all that stuff.

Hsu: Could you talk a little bit about your work on PUB? With Les Earnest?

Tesler: Sure. I took a year off—I didn't know how long I was going to be taking off, but it turned out to be a year, from the computer industry to do a very counterculture thing, which is to go out in the country and live on a commune. Well, it wasn't really a commune, technically, but it was that sort of thing. Six friends got together and went in and bought a property, 80 acres for thirteen thousand dollars in southern Oregon and tried to make a go there. Well it turned out after we got our houses built and all that sort of stuff and running out of money there wasn't really any economy there to—I went into a bank to see if I could get a job as a programmer and they said, "When we have an opening in the software department we give it to one of our tellers and train them to learn how to—learn how to program." So I thought, well, I guess I need to go back to the Bay Area at least once in a while so I can do some programming and make—pay the bills. And when I did that I ran into one of my friends at the A.I. Lab, and he said, "Oh, you're back. Alan Kay is coming by all the time asking where you are and he wants to talk to you about something." I said, "Well, what?" I mean I barely knew Alan. And he said, "Xerox is starting a research center and Alan is going to running a group there, and he wanted to—he recommended you for the center in hope that you could—you'd be interested in working in his group." So he was not available at the time, he had left town for some personal reason but I went over to PARC and I said, "Alan Kay said I should be talking to you guys." And they said, "Oh, yeah, we think that you'd be the right kind of person to be helping us here." And I said, "Well, can it be a part time or intermittent kind of job because I have this land in Oregon I've got to take care of too." And they said, "No, we can't do that, you have to come here full time and work." So I went away to think about it, decided, well, this is actually a pretty exciting thing to do, more exciting than trying to start a commune in Oregon, and so I went back and said, "Okay, yes, I'm interested." Oh, there's a hiring freeze at Xerox. It just went into place a few weeks ago and it'll go on for several months. It's good in that we didn't have to cut our size, which was tiny at the time, like eight people or something. But we can't really—can't really hire you. So I went away and came back, and went through a couple of round of interview, offer, reject offer, caused total commotion at Xerox because no one had ever rejected an offer before, and I thought—I thought they were undervaluing the experience I had running a company and just wanted me to—oh, and I didn't have a PhD, and so they viewed me as one of the coders. I was just going to code other people's ideas. And I went, now that's not what I thought was going on here. I have some ideas too. And so it took me about a year and a half to go through this process of finally getting to work at PARC, and they had to compromise, I had to compromise, and later I regretted the delay because there were lots of early decisions they made, even things like layout of the keyboard on the Alto, but more important things than that that I could have been involved in the decision and didn't get to weigh in for several years after that because I wasn't there when it mattered.

Hsu: Oh, we were talking about PUB.

Tesler: Oh yeah. So PUB. When I came back and didn't get the job at PARC, the hiring freeze thing, I went back to the A.I. Lab and I just went to John McCarthy and said, "Is there anything I can do here that's not artificial intelligence?" And he said, "Well, you should talk to Les Earnest," who was the real guy that ran the A.I. Lab. McCarthy was the—was the figurehead, and Les really just made things happen, and he was very interested in typography and in what became Unicode, in the problem—there was no Unicode yet, but the problem that solved, he was extremely interested in it. He even started a company while he was still at the A.I. Lab to do a microfilm printer sort of thing from a computer, and but he offered me a choice of four projects, and they all were supportive of the A.I. mission but not A.I. in themselves because I wanted to get instant gratification, not okay, we'll do a series of these things for the next thirty, forty years, and then we'll have something that's interesting. But I want to do something people can use right away. So he told me about his interest in typography. I said, "Oh, wow, I'm really interested in that too." He said, "Well, then maybe you would like this idea."

Tesler: His idea was to have a compiler that compiled text and other printing objects, publishing objects, pages, indexes, tables of contents, cross references, I mean whatever I could do. And being a little naïve, "I'll do all of those." That sounds easy. I've just been trying to do A.I. for years. This has got to be easier. So and in fact it was easy. He basically laid out how he thought it should be done and I did pretty much that, which was to take a language that was like the SAIL language, eliminate everything but text strings in terms of data types, find some way to handle numbers when there aren't any numbers, only text strings, and I did something that people today would recognize as not the same syntax but the same idea of—as HTML, JavaScript, and maybe even a little bit of Style Sheets, and it was a very—put up in competition with those other technologies it was pretty crude, but it hadn't been done before, to our knowledge anyway, and it was adopted right away across the ARPANET at the ARPANET universities who that—if they had a PDP-10, which was about half of them. So many theses for a couple of years in there were published using PUB, the document compiler [people] called it.

Tesler: And one month when I was ambitious—about fifteen years ago, I think—I went through the entire PUB manual and OCRed it and edited—actually annotated—that. Anyway, I—oh no, it was image. It was images of the PUB manual pages on the left side and then on the right side there was my commentary, which—and the commentary included "this anticipated Microsoft Word and in Microsoft Word you do this, and in PUB you do it this other way, or in JavaScript you do this, and in PUB you did it this way." And there were parallels in—mostly in those two technologies, Word, and JavaScript and HTML. And just again, it was not—it was inferior to them in the details of it, but it was very powerful, and you could have—you could—you could probably implement an HTML processor in PUB, and it was a powerful language. But then I left to go to PARC and just abandoned PUB and so it was unmaintained, and people were pretty unhappy with me because there were still a few bugs, but it was open source, and so other people actually went in there and found and fixed bugs eventually.

Brock: Was it to Les Earnest that the idea of kind of creating a programming language that would compile as a printed page was that—

Tesler: Yes.

Brock: His idea? That was new from his mind.

Tesler: Well, there were things around. There was troff, runoff, the other similarly-named technologies, mostly at MIT, that were early markup languages, very crude markup languages, and to the point where everything was either a line of text that you usually would render verbatim, or it started with a dot at the beginning of the line, and dots don't really belong at the beginning of lines, and if there was a line with a dot that was a command, dot, command name, and then parameters, and that was it, that was runoff, and it had maybe 25 commands or something. But we were both very familiar with that and some of the other things that were brewing, and he kind of jumped as far as he could think of being a person who worked in an A.I. lab and said, it's a general programming language for text as the—as the principal data type, and just like JavaScript eventually did or HTML with JavaScript, every string, if it was all digits and periods and commas in just the right places, it could be interpreted as a number, then you were allowed to do math with it, and if you had a—yeah, well, that's it basically. You could do math with it. And so nothing was stored as a number. We just stored everything as a string. But yeah that was I'd say his idea, that was the first I ever heard of it, but it was built on the knowledge of these very simple runoff-like languages.

Brock: May I ask one more question, sorry Hansen. How does it connect to Donald Knuth's TeX, and also how does it connect then to this whole interesting line of things at PARC of JAM and Press and Interpress? These—to my naïve view there's a family similarity going on here.

Tesler: Well, yes, and one thing I didn't say was the—not only did PUB have a few bugs, it had a few usability issues of its own, and I have—I can really only blame myself, but it isn't what I designed. I designed something that was—where the begins and the ends were matched, and you could say, center … un-center, that sort of thing, or—I can't remember all the details of it, but everything was parsimonious—naming, and one day I got a call from someone who I didn't know saying, "I hear you're working on [a] publishing language. So am I. Can we meet and see if we can do anything in common, make a standard or something?" And I was excited about that, I thought, yeah that's great. So we came down—and *he* came down from Berkeley and his name was Paul Heckel, and we—I showed him what I was doing, and he told me a little bit about what he was doing. He was not as far along as I was, and wasn't planning on doing as much as I was. And so he said, "You know what I'm doing is kind of a subset of what you're doing." And I said, "Fine, just take what parts of it are interesting to you and put it in your thing." He said, "Well, I'd like you to change what you're doing." I said, "What do you mean?" And he said, "Well, I was hired on a contract to do something for this guy and he's a little unhappy with my progress and I don't really have time to go back in there and change the names of the commands to be like what yours are, but you could change the names of the commands to be like mine." And I said, "Well, I guess so, sure." But I hadn't looked at it yet in that kind of detail. When I looked at it I went, oh, no, there's no parsimony here at all. The start of something and the end of something don't have any recognizable connection. And so I complained about that to him, and he said, "I really can't do it, Larry. And it'd be

great if this came out and everybody used the same thing, it would become a standard and just like Fortran it has its warts, but everybody uses it." So it was a tough decision. I made the wrong decision, I would do what he asked. After people started using it they came to me and said, "Larry Tesler, how could you make something so unusable." And I explained the situation and they said, "Well, that's ridiculous, we know Paul, and you should never have done that for him." And so I contacted him and he said, "Oh, that project was cancelled." I said, "You didn't tell me it was cancelled." He said, "Oh, just like—it was cancelled like two weeks later, the guy was real unhappy with what I was doing." I said, "You didn't tell me." So by that time I was out of there and over to PARC, and I couldn't go change anything, and it was a mess, but the—what was the question?

Brock: The lines to Knuth.

Tesler: Oh, yes. So one person who was—who loved and hated PUB was Don Knuth. And he loved the idea and he got very excited about it, he thought he could use it to format his books. Again, he's another typography nut, more than the rest—more than all of the rest of us put together, and he was trying to do something in that realm, and just started initially with PUB because it was powerful, but the more he used it the more he hated it, and he imagined what a language would look like that would be more appealing to him, and it wasn't like it was the PUB that I almost got out and should have gotten out, it was different. He brought his whole own spin to it. There was also another person at Carnegie Mellon, Brian Reid who did something called Scribe, and it was done also in reaction to PUB. He used PUB, he tried teaching PUB to secretaries at Carnegie Mellon Computer Science Department, and wasn't able to. And so he made this simpler language, implemented it in PUB, and they were fine with it. Then he made a native implementation of Scribe and it became very popular. So the other thing about PUB was that it used the whole character set of the A.I. Lab keyboards, which was something like 200 characters, none of which, outside the ASCII subset, none of which were standard. Nobody else ever adopted [it] outside of—outside of [the] A.I. Lab, and so nobody could use PUB unless they had a SAIL keyboard and a PDP-10. So PUB was a hit for about two or three years and then it just died because of all these kind of problems. Let's see. I think I'm not—still not answering your question. Is that close enough?

Brock: Well, the second part was just about how it connects to—

Tesler: Oh, other things at—

Brock: At PARC.

Tesler: At PARC like Press. JAM I'm not familiar with but Press was happening when I was still there, and but that was not—I guess in some sense you could think of Press as being in the same category as PUB, but I don't think it—I don't think we did. And I'm trying to think of what difference, what was a fundamental difference?

Brock: Maybe the graphics?

Tesler: Yes, yes. It was—they had to do graphics because they were going to do printing, and PUB was much more—much less ambitious in terms of print quality, typography, graphics. It just was very focused on the overview of the—of a book, the structure of a book, and anything that was a graphic would just be space left on the page for somebody else to do manually somehow, get that in there, until there was going to be cut and paste of course. But to me that was not—the world wasn't ready yet for cut and paste. The terminals weren't sophisticated enough to support the—WYSIWYG.

Hsu: We also discovered something on Les—a webpage that Les Earnest wrote that's been archived by the Internet Archive, and he claims that SAIL had something called "Pieces of Glass" that prefigured [overlapping] windows, windows that were done at PARC. So could you talk a little bit more about that?

Tesler: I vaguely remember Pieces of Glass. There was such a thing, and it had—I'm not sure I ever used it, I think I was gone by then. But remember Alan Kay was there. So it's hard to separate all these things out. People shared ideas with each other and so I don't really—I'm not the one to ask.

Hsu: Well, that's a good segue to talk about Alan Kay so let's maybe move back in time a bit to when you first joined the staff of SAIL around the time that Alan Kay also joined. So what was your first impression of him and your first interactions with Alan Kay?

Tesler: Somebody told me Alan Kay was there, and I said, "Who?" <laughs> It was that kind of thing. And when Terry Winograd first showed up his reputation preceded him, but when Alan Kay first showed up, at least from my point of view I just hadn't run into the name. But my colleagues had, and they knew who he was, and he was there for a certain amount of time, I don't remember, it was order of magnitude, a year. And it was like a visiting researcher sort of role, and he was intrigued by one project that we were doing that—but I keep talking about my friends at PARC—I mean at SAIL. They also both worked at Apple later. It's Horace Enea, E-N-E-A, and David Canfield Smith, and they knew Alan right away and we were—they were working on a declarative language based on rewrite rules. Today, wherever there's chatbots there's things like the Mad Doctor. They—well, it's one approach to natural language processing and other things. You can—what Dave Smith and Horace Enea wanted to do was kind of technology-driven. It was—if the problem could be solved using rewrite rules, pattern matching basically, then they were interested in the problem because it needs rewrite rules, or rewrite rules are a way to make it work. And I mean that was kind of—they were working for Colby, and he—what we now call chatbots is what he was working on, and so Colby welcomed anything that would make that a more powerful system because he would benefit indirectly. And so Alan was interested in pattern matching as a way to have object-oriented programming, because he'd already—he already was interested in object-oriented programming from Simula, and in his view you could have asynchronous processes, parallel processes that you would program, and you would—it would be a—he was very taken with message passing operating systems, and so in Alan's mind he found a way to put these things together so that they created a useful whole.

Object-oriented programming, pattern matching, message sending, he got them all in his mind as being all facets of one thing, and his application area in those days and many—and for most of his life has been education. Applications of things like teaching programming to kids; that's one of his beloved goals. And all the Smalltalk and similar spinoffs have been in pursuit of that goal. So he looked at what we were doing and we had probably the most elegant rewrite rule syntax. It wasn't that different from others, but just in certain ways, it was cleaned up and Dave Smith was very much like me in his passion for usability and parsimony, and he and I were almost competing on who could make it simpler.

<laughter>

Tesler: And—

Tesler: We published a paper—no. We wrote a paper and I think it was published. Yeah. We wrote a paper and it came back from the reviewers with, "Well, there's nothing new here."

<laughter>

Tesler: "It's all very trite. It's not clear what the applications are, but it's so elegant—"

<laughter>

Tesler: "—that I'm going to approve it for publication this one time."

<laughter>

Tesler: And it was just a beautiful thing. So—and—

Tesler: But it—we didn't follow up well enough and everything we did was just aimed at short-term Colby goals. But for years, decades, <laughs> Dave Smith has continued to maintain an implementation of this language. It was called LISP70.

Tesler: And he's retired now, but I bet he keeps it running. <laughs>

Hsu: So what did you make of Alan Kay's ideas at the time and—did you share any common interests or influences?

Tesler: Let's see. Well, everybody was influenced by Engelbart if they were into interactive computing.

<laughter>

Tesler: And Ted Nelson. They were the prophets and—

Tesler: —Alan no less than anybody else—

Tesler: But—

Tesler: He was—well, the few times that we interacted at PARC [he means at SAIL], he was clearly brilliant and he would understand immediately what it was we were doing and then run off to—way ahead of it—

<laughter>

Tesler: —in the dialog with us. And so, it was always an enlightening experience, but it only happened a few times in that context. I got to know him, really, when he went to PARC and he wanted to hire me, but didn't have the headcount. So he had this trick that he played, which is he'd find people he wanted to hire, he'd get somebody else to hire them on their headcount—

<laughter>

Tesler: —and then, some—and then, just count on the fact that they would come over and visit a lot and share their ideas and contribute. And so, I was doing that. I was contributing a lot, but when the project that I was helping on clearly was not going anywhere, it was going to be obsoleted by the Alto, basically, there was this window of time where I was a free agent.

<laughter>

Tesler: And so, I said—I told my boss that I'd really rather work on Smalltalk with Alan Kay than work on office information systems, which is what I was asked to do initially. But that was being managed by Bill English, who was very much of a—

Tesler: —make it happen kind of person. He would just get the material realization of the ideas people had. That's what he did at SRI. He tried bringing some of the—some of his team over from SRI and get

them to do similar things. But Xerox, as a company, was conflicted about what this was all about and whether they should be doing it or not and—

Tesler: And the different groups at PARC were friendly at the surface and cooperating and—but also inevitably, they were different ideas and the ideas were competing. And—this was a case where the vision that Bill English and his team had was ahead of anything that could be done at that time. They were—they had—and actually, the same for Alan Kay. Alan Kay wanted flat panel displays and color and things we just couldn't do. Bill English wanted distributed operating systems and things that we now can do, but, back then, it was just a little beyond the capability of technology in those days. Anyway, I found it very frustrating to work in the group. They had me assisting other people and this is—Alan had a different attitude and any idea I had, he was always, "Yes. Try it. Sounds like a great idea to me." And he motivated his team by empowering them to—empowering us to try things. If an idea sounded great, let's have somebody work on it for a reasonable amount of time and see what comes of it.

Hsu: Right. So the—Bill English's team working on POLOS?

Tesler: Yeah.

Hsu: What, specifically, did you work on on the POLOS?

Tesler: Not much.

<laughter>

Tesler: That was the thing. He was—he liked everything I did. That was the thing. It was, again, kind of like LISP70, I guess.

<laughter>

Tesler: Bill, he was a great boss to work for in the sense of being able to talk to him about anything that's on your mind or whatever, bringing him new ideas. But when I first got there, he said, "For the first couple of months, work with this guy," who's also started at the same time as I, Jeff Rulifson, and "Just come up with some ideas, and then, we'll get you involved deeply into the details of POLOS." And that was a very smart move. He knew I would just, for sure, be totally bored—

<laughter>

Tesler: —if I just went straight into POLOS. And he wanted—he knew that I needed to have some of my ideas in it. So Jeff and I got together and did a vision for the future. We called it OGDEN, the Overly General Display Environment for Non-programmers, O-G-D-E-N. And it had cut and paste—this is—didn't have anything. This was a vision document, but it would—there it was, the phrase is in there, cut and paste [OGDEN Overview, April 26, 1973, p. 12]. And it had fonts [*ibid*, p.11] that were decorative as well as functional and it had icons [*ibid*, p. 5], and that was because he was reading a book about symbols, signs, and icons. Just—what is the name of that field? Semiotics. He was reading a semiotics book and [it] defined the word icon. He said, "We'll have those." And so, we had a great collaboration. But then, we drifted apart and he got involved in POLOS in some mundane area. I don't even remember what he—what his job was in POLOS, but I think what they were thinking of for me was when it got—when the user interface came up, they were going to want to make it usable and they had me there for that day that it became testable. I would then start testing it.

<laughter>

Tesler: You know, sort of thing. Or maybe I had suggested that, whatever. It was just—and so, sure enough, I wanted to do a usability study. I hadn't done one since I was running my own little company. And—but I knew that the people at PARC were much more sophisticated than I and they probably knew the right way to run a usability study. And, in fact, they did and there were these two people that showed up from Carnegie Mellon called Tom Moran and Stu Card, who were with Allen Newell, and they had developed similar techniques and much more based on cognitive psychology. And so—

Tesler: We showed each other—we actually demoed for each other the running of a usability study and they said, "We have somebody who does this full-time, basically, that we've—we're hiring her and we're going to bring her out from Pittsburgh and she's going to do this. She could—she can run usability studies on your project, and so, she did. But before that, I ran a few myself and one was on a piece of the text editor part <laughs> of POLOS, which was just by default sort of like what Engelbart had, in NLS. It was very similar, but simplified. But not simplified in a way that made it a lot easier to use or easier to learn, just more—easier to implement and narrower in scope. It was basically just a way to edit source code (as opposed to any kind of document) and documentation for source code. So it was that kind of thing, which is what—at that point, people had decided that's what NLS was good for and—good for programming code and for programming documentation. So I ran a [two-day] usability study with a brand new secretary who had very little experience in word processing and it—the POLOS thing just bombed. I mean—

<laughter>

Tesler: —couldn't—she couldn't do anything [without getting lost in the mode tree].

Tesler: Before that study, Barbara Grosz (then Deutsch) had helped me to run a blank screen experiment with the same secretary. We said, "Okay. Just imagine—" I blanked the screen and said, "Imagine there's text on the screen like there was in word processors that you've seen before, but you—but it works a different way. You don't edit the same way. You edit in a more intuitive way." And so, I basically let her design it and I said, "How would you tell it to delete three words from the middle of the page?" And she'd say, "I'd point at the screen and I—at that first word and then I'd sweep across to catch the other couple of words and then, I would hit the delete key on the keyboard." All right.

<laughter>

Tesler: She didn't say start by hitting the delete key on the keyboard. She started with the object and then the command. That's good. That conforms with this theory I've been doing, which is that modes are a bad thing for text editors because, in those days, people used to transcribe a lot. And so, they're looking at a document, retyping it, and they can't look at the screen and see that when they—that when they hit the end of the delete command, they forgot to—they didn't hit the terminator key properly and now, it's—they're deleting their whole document and everything else on the company's server or whatever.

<laughter>

Tesler: So she designed it and any time there's a PARC reunion, I run into her.

<laughter>

Tesler: I remind her—

<laughter>

Tesler: —that you designed the graphical user interface.

<laughter>

Tesler: But it was—she basically validated what we were trying to do.

Hsu: Hmm. So you had already—you already decided from when you saw the NLS before that modes were a bad idea, but you hadn't had empirical confirmation that it was a bad idea until this point?

Tesler: Whenever I said—whenever I learned a text editor or other kinds of software in those days, there were usually modes, and modes were thought of as a good thing, and they were a good thing in a lot of ways. If you—if we—if you say “move,” then you know the next thing to expect is going to have to be “go somewhere else.” And so, you can probably find some interesting algorithm to take advantage of that. And—but if you just do a selection, there's not a lot you can do in preparation to make the next thing that happens fast, because we don't know if you're going to delete it, move it, duplicate it, whatever. So it was well known that modes are a bad thing. I only saw one other publication about it, ever. Somebody wrote something on why modes are bad or something and—

Tesler: But, to me, it was a problem that could be solved.

Tesler: And so, I started just, in my notebooks, it's, like, what is a mode? What situations put you in a mode? What is—does the syntax of the programming of the—I mean, of the command language—

Tesler: —in its—by its very nature of the syntax, does it make modes happen? And, I mean, these are all very fundamental questions that I was trying to answer and—

Tesler: And there were other people who kind of griped about modes a lot, but very few people did anything about it, as you—“Everybody gripes about modes and nobody does anything about it” is what I used to say.

<laughter>

Tesler: And so, in the case of PARC, having that testing of the NLS-like editor, yeah, validated what I was doing. But it wasn't the first time that I was getting involved with trying to ban modes. I was trying to ban modes for years—

<laughter>

Tesler: —and had to eventually give into the fact that there are things that you do on computers that work just better with modes. And—but how to make that appealing to people and less error-prone, instead of trying to just eliminate them altogether.

Hsu: Hmm. I see. We were talking about how you were working on POLOS, but then, you were more interested in working on Smalltalk with Alan Kay during this period where you were sort of, I guess, a free agent within PARC. So could you talk about what you were doing for Alan at that point in time?

Tesler: Yeah.

Tesler: Several things. First of all, we were designing the language and that's something I had some experience [with] and Alan knew I had some experience. I had redesigned this animation language for art students, of course, and I had also worked with Dave and Horace at the AI Lab to design some LISP70, and PUB was another language I'd worked on. So he wanted me to help with the language design, but, mainly, that was Dan Ingalls' job and—but several of the rest—several other people in the group helped Dan, basically, but it was really his accomplishment. And—

Tesler: In fact, in preparing for today, I thought I'd just skim through my files of Smalltalk stuff in my garage and came to a place where I had notes on meetings. I think they were meetings with Dan about what the syntax should be for Smalltalk-76 or something like that. And—

Tesler: So I got involved a little bit there. I got involved a little bit in implementation and—but mostly, I did—I used Smalltalk and, interesting, my files at home have—for each year, they have a file on [the] system and another file on the user interface. So what I did that year, and in some cases, wasn't—what's in my files may not be what I did. It could have been what somebody else did and I have a copy of the memo. Let's see. I contributed to what became BitBlt. It was—I was doing a very early text editor, didn't even have cut and paste in it yet and it didn't even have that functionality, I mean. So it didn't have those commands either. And—

Tesler: I wanted to have a character that blinked between white on black and black on white, whatever—I think that time, we had green screens, but—

Tesler: And the Alto did not have a way to blink a particular character. But, actually, it wasn't the Alto, it was the POLOS hardware, which was on a Data General Nova, and the screen didn't have that capability built in. You had to add it. So I did that, but then I started thinking about the Alto and whether—and what kind of primitive operations would be useful in implementing things like scrolling and highlighting of selections, and things like that. And the highlighting of selections, the inversion of the characters was useful, but for things like scrolling, taking a big block of stuff on the screen and moving it someplace else on the screen by doing a memory copy, would be a way to implement scrolling. And—but I knew that proportional fonts were coming. At that point, we just had monospace fonts and it was going to have to do arbitrary rectangles and shapes that weren't quite rectangular, but, at the moment, I was just going to—my idea was that we would just do rectangle operations. So I had something called RectOp and went to Chuck Thacker, the main hardware guy on the Alto, and said, "We really should have a microcode to implement something I'm calling RectOp." And so, I explained the kind of things that it could do and he said, "That would take 300 to 500 words of microcode." I think he said 300, and it turned out he was right. It was—ended up being 300.

<laughter>

Tesler: Three hundred lines of microcode, or something, and we just can't afford to do it on something as trivial as graphics.

<laughter>

Tesler: And—because there wasn't room in the existing Alto, but for the next Alto, they were adding 500 instructions. So I thought this is great, 300 out of those 500 could be graphics. And at first, he said, "Nah." It was too specific for an application and he didn't want that to be what the Alto was. Well, other people talked him out of that, but the main person who did was Dan Ingalls. Dan decided that he would take the RectOp idea. He had some great ideas for making it way better and more powerful and I had things like move a block of bits from here to there, based on their X,Y coordinates, and he would say, "Oh, but there would be a mask or a filter or something applied." I went, "Geeze, I've never done that." That was a kind of graphics I hadn't done. <laughs> So he had some other experience from—in his life doing graphics and that came to bear on his version of RectOp that he called BitBlt, way better and way more powerful. But he would agree that he got the idea from me, but just took it to a different level. And—but the really—the thing he did really well, <laughs> which I learned a lot from, was he implemented it and he knew that, well, he had to learn to do microcode to do that, and that's what stopped me. He said, "Why don't you implement it?" to me, and I said, "Oh, I got to learn microcode and bla bla bla." And—

<laughter>

Tesler: And so, he said, "Well, I'll learn microcode." So he went and learned it and did it, but he was smart. He implemented it in—BitBlt in Smalltalk, ridiculously slow, but functional; used that as a way to get it right, what are the right features and what algorithms and so on. Once he had that down, he rewrote it in BCPL, which was the assembly language kind of thing. It was C. It was more like a C language. And so, he translated it from Smalltalk to C, BCPL, and then, he wrote it a third time in assembler and then, a fourth time in microcode.

Brock: Oh my Lord.

Hsu: Wow. <laughs>

Tesler: Total, a month.

Hsu: Wow.

Tesler: And I think it was around a month and there was BitBlt with zero bugs, guaranteed.

<laughter>

Tesler: Because of the way he developed it, and very compact and he gave a few demos and Chuck Thacker couldn't say no. He said, "All right. Three hundred words."

<laughter>

Tesler: He did it in 300 words. I'm going to put it in.

Hsu: Hmm. Could you describe the relationship between Smalltalk and the Alto?

Tesler: Ah. Okay. Some of the—what I'm going to say, I figured out later.

<laughter>

Tesler: But all the information was there. I should have figured it out sooner. But—

Tesler: The—Alan Kay wanted to get some things done that he didn't have the resources to do and Butler Lampson wanted to get some stuff done that he didn't have the resources to do. So they got together—

<laughter>

Tesler: —and said what they were doing was for the other guy.

<laughter>

Tesler: I'm doing this thing. Dan Ingalls and Alan were doing this thing that could be used by the Computer Science Lab—we were in the Systems Science Lab. So it could be used in the Computer Science Lab for all the various projects that Butler Lampson was hoping they would do and giving Chuck Thacker a real good hard-to-do hardware design project. And they were doing what Alan wanted, [which] was the interim Dynabook. And so, it was kind of a—

Tesler: It wasn't just, Alan needs an interim Dynabook and so, we'll do that for him as a favor. It was they were looking for something that was—that they—that everybody could have one and they could use it to show what personal computing would be like when it was all done. And—but kind of in the guise of I'm helping the other group. That's my own theory about what was going on, consciously or unconsciously.

<laughter>

Tesler: Yeah. But I've—it's not too far off.

<laughter>

Tesler: Does that address your question? Yeah.

Hsu: Yeah. Let's say we take a little step back and talk about Smalltalk more broadly. So how would you describe Smalltalk to someone who has a very basic knowledge of computers?

Tesler: Hmm.

Tesler: It's a language that we hoped, when we designed it, would be easy for kids to learn. But—

Tesler: What was true of it is that using it, people have developed tools in Smalltalk that are easy for kids to use and <clears throat> through which they learn things, including programming. So it was kind of the—it's the—a meta use of it. It's a platform for things that could be used by kids.

Tesler: And—

Tesler: Because Smalltalk's never been a very well funded project, I think. It's—

Tesler: Anyway. Let's see.

Tesler: But the other way to define Smalltalk is by what it is technically. It's a dynamic object-oriented programming language and you could explain each of those things. So—

<laughter>

Tesler: Dynamic means that memory is allocated only as needed and—

Tesler: —memory is managed automatically by a garbage collector. And—

Tesler: It's introspective in that you can write code about code. You can change the Smalltalk system in Smalltalk and—at your peril, of course, if you make a mistake.

<laughter>

Tesler: But Alan thought that was important, that if a kid had an idea, [if] they wished it was a little different, they could do it. They could make it different themselves. And, I mean, some high school students did that and—but it didn't turn out to be what people were really looking for. But then, another way to define Smalltalk is if you know what Java is, then Smalltalk is what Java came from. And—

Tesler: It was—Java is Smalltalk with a C-like syntax and each version of Java becomes slowly more like Smalltalk.

<laughter>

Hsu: Fascinating. What were the most captivating features of Smalltalk to you?

Tesler: Well, I grew up in an era when computers were too small and too slow. And so, optimization was a very important thing. You had to make your program fit in memory, not always easy, or, at least, break it into pieces that can launch each other, that allow it to run. And—

Tesler: So—

Tesler: Captivating?

Hsu: Captivating

Tesler: Captivating.

Tesler: Ah. And so, one of the features of Smalltalk that I found captivating was—

Tesler: —how concise you could make a program and, therefore, how it—you can make it easy to understand because it's—there's not much to study, or easy to prove that it works.

Tesler: And so, a well written Smalltalk program is short. Alan Kay was one of several people, I think, John McCarthy was another, who refused to write a program that was longer than one page.

Tesler: And so, if you wanted to make Alan happy with what you were doing, you would find a way to take his latest idea, not only implement it, but implement it in a page.

<laughter>

Tesler: And so, to do that sometimes, we had to enhance the Smalltalk language a little bit.

Tesler: And so, for some reason, I find that—I found that very captivating.

<laughter>

Tesler: Let's see. What else?

Tesler: Well, at the time, object-oriented programming was new and so, the ability to program by inheritance. In other words, define an object in a very general way, and then specialize it in several derivative classes.

Tesler: Factoring the program so that the—you had a parsimonious set of classes that form a hierarchy based on inheritance was—I found very captivating and—

Tesler: Yeah.

Tesler: And then… yeah, I think that covers it.

Hsu: Sure, okay.

Tesler: Anything else I think of is kind of the same thing.

Hsu: Yeah. Earlier you mentioned—you were talking about prefix versus postfix. And Smalltalk syntax has very much of… you tell the object, you send the object a message..

Tesler: Yeah.

Hsu: …to do something. And you mentioned that you were—were you involved in deciding that syntax? Or by the time you got onto the project, was that already—that decision already been made?

Tesler: Oh yeah, the decision had already been made. And it was Alan Kay more than anybody, and if anybody else, Dan Ingalls. And one thing that… is also pretty captivating <laughs> I guess, that the people in the group really loved, was that object-oriented user interface was easiest to implement in an object-oriented language. And… because there are objects and verbs, and the verbs can be inherited and overridden in the code. But it also—it reflects the model of whatever the domain that you're working in, where the objects in that domain are the ones that you're programming. So one way to get people to appreciate object-oriented programming, is to show them object-oriented user interface. And how one leads to the other, and vice versa. If they're already sold on the object-oriented technology, inheritance, then you can say, "Well, look at how easy it is to implement this metaphor, basically. Because it follows the same pattern."

Hsu: Right. How important was the object-oriented programing paradigm to you? And how—were you particularly—did you find that this was going to be a revolution in software design? Were you sort of on that train from the beginning?

Tesler: Yes, I thought so. And when I got to Apple I had a project to do Lisa applications in an object-oriented language, and then Mac applications. And some were done in our language, but I was a purist about—from the Smalltalk world, thinking about what… an object-oriented framework should be like. And based—in Smalltalk there was a class called "Object" and everything else is a subclass of Object.

Hsu: <laughs>

Tesler: And so there's things that every object can do. Like report its class name, or maybe count how many of them are alive right now, export them to a file in a way that you can bring it back and kind of revive it. These are all things you could do with any object.

Hsu: Mm-hmm.

Tesler: So we had in MacApp, the Macintosh framework, we did the same thing. We had objects, and then we had different kinds of objects, like document objects, view objects and so on. But other people, at other research centers and development labs, had a very different idea. They liked inheritance and object-oriented programming, but they thought that getting it to the point where—that every object, even a number, was an object, did not appeal to them at all. They wanted to have, I think what we called it was "multiple roots." Instead of a tree, where everything is a subclass of Object, they wanted to have… several roots and not require that. And so I was a purist about it, and couldn't understand really what they were driving at.

<laughter>

Tesler: Until later. And their ideas won out over this single root idea.

Hsu: Oh, really?

Tesler: Yeah. And again, in Java there is an Object class, and there's a Class class, there are Number classes. But in most—well even people writing in Java sometimes do their frameworks in a multirooted way.

Hsu: Really?

Tesler: Yeah.

Hsu: Huh.

Tesler: I mean there's nothing that requires you to… have a single root. I mean it's…

Hsu: Oh, is it because they use interfaces? Is that how they get around it? Or they just don't subclass the root object at all?

Tesler: I'm trying to think of an example… but the answer to your question was yeah, they don't subclass the root object. And you won't find an API—in their API, if you go to see what Object does, it doesn't <laughs>. It just… it doesn't have a home in their API. But I think it comes from not having an object-oriented language to start with, they're starting with… just ordinary C or something.

Hsu: Oh.

Tesler: And they're… well it's been a long time <laughs>. Yeah… yeah I guess it's hard for me to support—I'd have to go think about it and come up with some examples. But all I know is that… is when—there are people that have told me they're using object-oriented programming, but they aren't using these strict hierarchies based on class Object.

Hsu: That's interesting.

Tesler: And then they explain why they're not, and usually it's because the language they're using is not fundamentally object-oriented. And they're using it in an unintended way.

Hsu: Okay.

Tesler: And they can get away with it with, say database records or something, but not with numbers. So…

Hsu: Right, I see. Because I'm just trying to compare it to my experience with Cocoa on Mac OS X, which… everything does inherit from the root NSObject…

Tesler: That's because it all…

Hsu: Yeah.

Tesler: …because Objective-C was based on Smalltalk.

Hsu: Based on Smalltalk, right <laughs>. So there's a natural… yeah, evolution there. Yeah.

Tesler: But say C++. C++ does not have—what it really lacks is introspection, it doesn't have introspective objects.

Hsu: Right.

Tesler: And so... yeah, so there's nothing like a single root.

Hsu: Right, okay. Right.

Tesler: That's probably the best example, C++. Because the language doesn't support it, so you don't even try.

Hsu: Right, because the root object allows you—provides those introspection capabilities. But if you're not going to use them, you don't need them.

Tesler: Yeah, right.

Hsu: Right.

Tesler: They design frameworks that don't need a root object, yeah.

Hsu: Right.

Tesler: If it's model-view-controller then they might have models, views, and controllers, but nothing that is a—that they all inherit from.

Hsu: Right, okay. Yeah. Sorry we got a little bit on the technical…

Brock: No, no. Fascinating.

Hsu: …into the technical weeds there <laughs>. Going back to PARC. So you were in Alan Kay's Learning Research Group for a brief period—or were you…

Tesler: I was actually probably there longer than I was—for a while I was sometimes allowed to go visit Alan's group, go to some of his meetings.

Hsu: <laughs>

Tesler: Then I…

Hsu: So you were in the POLOS group first, and then you eventually…

Tesler: Yeah, first I was in POLOS. Then I was kind of this free agent for a while, helping various different groups… including Alan's. And then I had my own project that was… yeah, after—there was a project called Gypsy that Tim Mott and I did…

Hsu: Right.

Tesler: …that was the first practical implementation of modeless editing, of cut, copy and paste. And like two hours after we <laughs> demonstrated that, Dan Ingalls had it running in Smalltalk, only better.

Hsu: Wow.

<laughter>

Tesler: He didn't have to—well sometimes it's agonizing to make design decisions. Because everybody's got an opinion, and a lot of them are trying to… prove that your idea's just totally bad anyway. So at PARC people were merciless about critiquing each other's approaches to things. And so Tim and I had agonizingly come to this set of user interfaces that… kind of a little like in PUB, where I had to make some compromises I didn't like. Tim and I had to make some compromises we didn't like. Because people were so sure that this couldn't be the right answer.

Hsu: Huh.

Tesler: And so…

Hsu: So you caved to pressure, or critiques from other people…

Tesler: Yeah, from other engineers. And... like double-click. Double-click was invented by Tim Mott. We were trying to figure out a better way to select a word, and I mean now it's so obvious, but nobody had ever thought of this. Well, [so] we thought. So he came in one day, he said, "I've got it. Double-click."

Hsu: <laughs>

Tesler: I thought about it for five seconds, I went, "Oh, of course. But… but…" And then I started doing like all the engineers do, "But this, and that, and… what if the mouse moves after the first click? What if the mouse—what if you wait too long before the second click? How do you know it's a double click?"

Hsu: <laughs>

Tesler: And so I challenged him on that. But I wanted to get to an answer, so that argument probably lasted five minutes. But every other engineer at PARC had an opinion about double-click. And some of them had thought of it before, and tried it and realized there were all these problems about, how do you define it properly? And what do you do if somebody makes a mistake, and they go too slow? Or… you know. So… I think initially we had—it had to meet a distance rule. Mouse couldn't move this much between first and second click. And also a time rule. But we weren't sure what they should be, and whether we needed both or—And so we did some experimenting, and eventually figured it out. But… there was probably a version in there where it was overprotect—we were overprotecting it. And not just doing whatever first came to our heads <laughs>, right? This is one we probably should have just done that. But we didn't want to be criticized. And so… we wanted people to go, "This is good enough." Right?

Hsu: Mm-hmm, right.

Tesler: So probably a better example of that is... how many mouse buttons. We both thought that there should be just one mouse button. And that way we could have touchscreens, where you just have a touch. Because that secretary just pointed at the screen and said, "I'm pointing with my finger at the screen, I'm touching the screen." And so we decided we would not use any extra buttons that were on the mice. PARC had two-button and three-button mice. And we said, "We're only going to use one." But it was so tempting to have, "Well…"

Hsu: <laughs>

Tesler: And somebody said, "Well, you know, I think this is going to be hard for the editors at Ginn and Company to use, because they have to learn that the—because they have to learn what this key does." I can't remember what it was even that it did. Oh, I remember. Yeah, it was "select word." If you hadn't learned yet about double-click, and it was too hard for you to do—we had this idea that it would be too hard.

Hsu: <laughs>

Tesler: Because everybody's telling us it's going to be too hard, because of the time and distance thing. Then you can enable a—you can use a different version, which is the second button on the mouse clicks on a word, and it will give you a wor—it'll give you a word. Second button. Why did we do the second button instead of double click? Because everybody was arguing with us that it was ill-defined. So we caved in too early. And later it came back. But we had this idea that they would say, "Well… there's these extra buttons, you're not even using them."

Hsu: <laughs>

Tesler: And, "What's the problem?" So anyway, we went through that a lot. Because we didn't have the PhDs, they had PhDs, you know, it was that kind of thing. But then eventually we got our courage up, and usability studies up. But Dan Ingalls just took one look at it and said, "Why do you have that second button doing double—you can double-click, but only with the second button?"

Hsu: <laughs>

Tesler: I mean we had double-click, but it was only with the second button. So the time and distance didn't matter, kind of thing. He said, "That's ridiculous." And he just implemented it as a simple double-click. He had cut, copy, paste, undo in a little menu, and that was all the commands you needed. I mean he just like, in minutes, he just made it—he got rid of all the wrinkles, made it easy to use. Easier than we had.[3]

Hsu: Mm-hmm. So the version of Gypsy that you and Tim Mott had created, was based on the Bravo codebase?

Tesler: Bravo codebase and... that was the only—Gypsy was the thing that Tim and I did. Dan was working on Smalltalk…

Hsu: Right.

Tesler: …in the Smalltalk codebase. And nobody was—he didn't have any promises to anybody that this was going to be hard or easy, or who it was for, or anything.

Hsu: Right.

Tesler: He just did what felt right to him. And it felt right to us too. So…

Hsu: So when—at the time you and Tim were working on Gypsy, were you actually part of CSL at the time? Or…

[3] Larry Tesler adds: Another example – Tim and I included *Undo* in Gypsy's spec but—for obscure technical reasons—we didn't implement *Undo*, even though Gypsy was built on the Bravo codebase, which did implement it. In Smalltalk's editing menu, Dan included *Undo*.

Tesler: I was never in CSL.

Hsu: Oh, okay.

Tesler: Gypsy... so that was when—I told you that Bill English had given me this sort of free agent sort of time. One day he called me in his office, he said, "We just had a visit from one of our subsidiaries." And he told me all about Ginn. And I go and, "Yeah, that's great. That's the kind of place that I'd like to be helping." He said, "Well, you're in charge of the project now."

<laughter>

Tesler: I said, "What project?" He said, "They came and they said every division of Xerox is taxed for research. And they're not getting anything out of the research. They want us to do something to do with publication as an application area." And so we got together with them, we defined a whole series of projects. Did the first one, never got to the second one.

Hsu: <laughs>

Tesler: Or maybe a little bit to the second one. Yeah, we worked a little on the second one, never finished it. And it was very slow, we did it in Smalltalk. The first project, Gypsy, was what they called the "manuscript system." We called it a typescript, but it was manuscript basically… or even a galley, it was a galley-making system, galley proof. Because it was just text, no graphics. But it was sort of half there, because you could have bold and italics and underline, but no other font changes. And then the second project was supposed to be the page—instead of the manuscript system, it was the page makeup system. And that's the one I was really interested in, because that's what would obsolete glue and X-ACTO knives.

Hsu: <laughs>

Tesler: And that's the thing I originally thought of cut and paste for. And so I actually did a mock-up of that. And it was a little bit like... on the iPhone or iPad, when you select something, it immediately pops up a menu, cut, copy, paste… and other things now. I had that idea in that thing I—it was called Cypress, it was the successor to Gypsy. And it was oriented towards page makeup, because you could make a selection, the same way you could make it in Gypsy. But when you cut and paste, it would make a distinction between pastes that filled—pastes that inserted themselves in the text, and reflowed the margins. Versus taking a blank space on the page, or a nonblank space, and plopping a diagram there or something. And Cypress could handle both situations. And PageMaker, which came a couple years later

from Adobe [actually it was originally from Aldus], was also—<clears throat> excuse me, based on this reflowing versus non-reflowing.

Hsu: Oh, okay. Meaning that text would wrap around graphics that you had in the document, or would flow around other things.

Tesler: Well it—the thing that was a little different about these programs, was that you had a choice when you pasted. Pasting… without affecting the flow of—without affecting the other text that's on the page, or other graphics on the page. Or you could choose to have it insert with, and push things to the next page.

Hsu: So… okay.

Tesler: Column two of this page to column one of the next page, for example. So we kind of had two different metaphors. The document as a long string of characters, versus the document as a collection of… one area per column, pages. Pages that have different areas. In other words, like PUB, it was PUB's model of the page. As opposed to Word's model of the page, which was text flowing.

Hsu: Oh okay, I see. So it's like you have two columns, and if you paste into the first column you have a choice of having it flow to the next one, or you would treat the two columns completely independently.

Tesler: Yeah.

Hsu: Okay.

Tesler: And so a background graphic, for example, you would just paste it behind the other stuff on the page. And it would become a watermark or a background scene, scenery kind of thing.

Hsu: Mm-hmm.

Tesler: But the thing that the iPhone did something similar to, was this thing where you automatically—as soon as you finish making a selection, without saying anything else, the command—or the menu, I'm sorry, the menu pops up of what you can do. And in Smalltalk, you had to push a menu button on the mouse for the menu to appear of things you can do. But in Cypress, just making the selection gave you this choice of what to do with this selection that you've made. And that's what happens on iOS these days too.

Hsu: Right, okay. So Cypress was the follow-on project to Gypsy…

Tesler: Yeah.

Hsu: …that you and Tim worked on.

Tesler: No, it ended up being just me.

Hsu: Oh, just you.

Tesler: The company decided that, "We'll make Ginn happy by having one person work on this, but what we're really about is office systems." And not POLOS, but this other, inspired by POLOS office system that they were going to do. And so a group of the engineers got together, as is done at PARC, people sort of self-form a team. And for a short time there was a team that I was leading, that was Cypress. But they all realized that the politics of the situation was that you get more credit to work on the office system, than you do to work on a publishing system for a little educational textbook company. Because after all, we're going to take on IBM and battle IBM pretty soon now. Anyway, so it was just me for a while with a contractor helping. And then when we finished it, what we were going to be able to do, we were way too slow, it was about 10 times too slow. So I got the PARC videographer to… make it a fast motion thing.

Hsu: Oh, right <laughs>.

Tesler: Take, you know, every 10 frames. And just… speed it up with the camera.

<laughter>

Hsu: So then… okay, so the office system that everybody was working on, was that the OfficeTalk system that ended up on the Star? Or was that something—a precursor to it? Or was it..

Tesler: Yes, the one that everybody else was working on was OfficeTalk.

Hsu: Okay.

Tesler: Tim Mott, Jeff Rulifson, Dan Swinehart, I think… And then that one also kind of fizzled away. But Dan Swinehart was left with… what do you call it? Woodstock, at some point that… yeah, for a while it

was called Woodstock. In 1975 there was an article in Businessweek, I don't know if you ever have seen it…

Brock: About PARC.

Tesler: Yeah. About IBM and Xerox, and the great competition they were about to have.

Hsu: Oh.

Tesler: And in there, they visited PARC. And they have—I have a copy of that issue of Businessweek.

Hsu: Oh, wow.

Tesler: But... they talk about Woodstock, and they talk about Gypsy.

Hsu: Hmm.

Brock: Hmm.

Tesler: And cut and paste.

Hsu: <laughs>

Tesler: That was the first time in a nonproprietary publication…

Hsu: Oh, wow.

Tesler: …that that was there.

Hsu: Oh, okay.

Brock: That was Businessweek, you said?

Tesler: Businessweek.

Brock: Okay.

Tesler: 1975. The Office of the Future was the cover story.

Hsu: Oh, okay. So the time that you were working on Gypsy and Cypress, what years were those?

Tesler: Gypsy was 1974 to 1975.

Hsu: So you had stopped working on POLOS in... ’72 or ’71?

Tesler: Well let’s see, I got to PARC in ’73, February, ’73. And... probably a year and a half I was in POLOS. And then Bill English realized I was unhappy, realized that Ginn was unhappy, and decided he could make us both happy…

<laughter>

Tesler: …putting us together. So that’s what he did. And then I started spending more and more time with Alan Kay’s group, eventually one—some quarter, because of budgetary somethings… I was officially in his group. For a while I reported to Mike Schroeder, who did Laurel…

Hsu: Oh, wow.

Tesler: …mail system, with Ben Wegbreit. And he had me come over and critique his user interface one day. And it was pretty good, to start with. But we made it a little bit better. But I found out later that he hadn’t really told anybody that I had helped.

<laughter>

Tesler: And... yeah.

Hsu: And that was before or after Gypsy?

Tesler: After Gypsy, yeah.

Hsu: Oh that was after Gypsy. OK. So then..

Tesler: So'75 was—Gypsy manual or something is dated April '75, I think it is. Tim Mott was going back and forth to Boston, to work with Ginn. And later... after I had already left the company, a couple of other people did a study at Ginn. They were still trying to use Gypsy, but not that successfully, because there wasn't anybody there to fix the bugs at Ginn.

<laughter>

Tesler: But I got involved with Alan's group, I got working on the NoteTaker.

Brock: Hmm.

Tesler: Now there it was more of a business contribution than a technical contribution. One of the programmers… Kim McCall, I think was his name… was looking for some—he was very young and above his experience level, doing this project which was called, I think TinyTalk, for the NoteTaker.

Brock: Right.

Tesler: And so they assigned me to be his mentor. But he did all the work, I just helped him understand concepts, and unblock him if he was getting stuck. And he got it to work. And then I did my first hardware project, I did a little hardware project on a PET computer one day at home. But this was kind of like the Josh Lederberg bet, you know? <laughs> This wasn't a bet, but..

Brock: <laughs>

Tesler: Doug Fairbairn, who you know well <laughs>, decided that any programmer should be able to do hardware. Hardware is easy, software's what's hard. And so he decided to start with me…

<laughter>

Tesler: …and see if he could—what gap there was between the state of the art, and what the state of the art would be if Larry, who never really did any hardware before, or took a course or anything like that, would be able to actually build a working board.

Brock: Hmm.

Tesler: So he decided it would be the processor board of the NoteTaker. It was a six-board or seven-board machine, I can't remember, and the main processor board—which sounds really fancy, but it was actually quite the opposite—this was the least interesting board there.

<laughter>

Tesler: But he wanted to see whether using the tools we had, which was SIL and the... what do you call that machine? The Stitch Weld machine.

Brock: Right.

Tesler: And a logic analyzer. And given the state of digital logic analyzers that year, and Stitch Weld, I was able to do it.

Brock: <laughs>

Tesler: Not only that, we sourced the first Intel 8086s, and got them first. First day I had to—my first Stitch Weld board did not work. So Doug came over to help me and, "Dah, dah, dah, dah…" He said, "The pins are backwards."

Hsu: <laughs>

Brock: Oh.

Tesler: So he went to the documentation and the documentation was backwards. It wasn't my fault <chuckles>.

Brock: Oh, my God.

Tesler: So we reported to them that the pin numbers were Big Endian on a Little Endian machine. They must've hired somebody to document it who came from the other world and so I just simply, in one minute in SIL, reversed the pins and worked second day. Second night, run, it worked. And then the other thing I did was some architecture on the Ethernet board. The Ethernet board had room for 25 chips. That was the budget, but the Alto Ethernet board had a hundred chips, so we had to figure out a way to get a

hundred down to 25. So because I was also a hobby computerist, I went to my Dr. Dobbs journal and started leafing through the pages and found somebody who implemented a—what do you call it? Wow. It's—basically, it's a hardware implementation of a finite state machine, but what is it called? Anyway, some standard circuit—and he found a way to do it in software, so turned out that this function was, like, half the chips all had to do with this one function. If I could do it in software, I could get rid of almost all those chips, and so—but the Alto software—the Alto hardware wasn't quite fast enough to—well, it was just fast enough to keep up with a stream of bytes coming in from the Ethernet, but it wasn't—also, it wasn't fast enough to identify the source—the destination port of the packet, and so I unwrapped the loop and just made it that it would—every other instruction would be making a further step on this state machine work. Anyway, yeah, it would—I'd be alternating basically a little work on the bringing in of the next byte, because if I don't grab it before it leaves the one—this tiny little buffer, I'd lose it, and then also deciphering whether this is really for me or not, and after about three instructions, I was able to figure out usually that it's not for me and just abandon it and save the time, but if it turned out it was for me, then I already had started grabbing the bytes and made it work that way. Technique—yeah, good software technique: unwrapped loops. Very important back in the day when things were slow, but now...

<laughter>

Tesler: Yeah.

Hsu: Interesting. So then—so the NoteTaker projected in—was it '78, 1978? Is that correct?

Tesler: Yeah, I think so.

Hsu: Yeah, and then I think in your previous interview, you also mentioned that you worked on the—sort of the integrated development environment for Smalltalk and that's where you invented the class browser in this case.

Tesler: Yeah, the browser.

Hsu: Yeah, so that was before the NoteTaker?

Tesler: No, that's—oh, probably was before the NoteTaker. So the class browser thing, wanna hear about that?

Hsu: Yeah. Yes.

Tesler: Just like with Engelbart, when I first saw the Smalltalk programming mechanisms, I was shocked about how primitive it was, and [the] first version was done by John Shoch, and he did it in kind of a very, even by that time, dated user interface, and kind of "why are we seeing this in Smalltalk?" It was very modey and—but it was functional. We just had to have something to develop with, and so soon enough, we developed things that were better, but Dan Ingalls had developed most of the stuff we were using at a certain point in time, and...

<Pause in thought>

Tesler: Let's see, what was the—oh, okay. Yeah, again, anytime it was—you could always have a terminal window, like the terminal window or the—on the Mac or the Windows terminal window, and there was a terminal window in Smalltalk that you could go into a read-eval-print dialogue and so the way that coding was done was by typing commands, so programmers were not given the same modeless cut-copy-paste interface as end users, and there's probably a good reason for that initially because—but you can—because you haven't bootstrapped yourself up to this level, but that's what we were all using was—if you wanted to, say, define a new method, you'd say "Class so and so, new method, colon, whatever," and you'd give it as a command, and then when they started trying to do better than that, giving the user a little more consumer feel to the technology, they didn't go very far. Anyway, there was a woman in the group named Diana Merry and she had a lot of meetings with Alan and Adele and—'cause they'd asked her to do some things, and she overheard something that Alan said, which I had heard also once or twice, which is, "No one's ever come up with a good way to browse on a computer." And so she started ending every meeting saying, "You know, we still can't—no one's ever come up with a good way to browse on a computer," and it became almost a joke that Diana would always say this 'cause then that was the end of the meeting. And one day, I said, "I've had it! I've had it! I've heard it enough times. I'm gonna come up with a good way to browse," so of course that's what she wanted to happen. Somebody in the group would volunteer, and so she taunted me about it. I said, "I am gonna come up with a good way to browse," so I went back to my office and I thought, "Browse what? What am I gonna browse? Browse books, browse databases?" I thought, "Oh, the programmers would appreciate it if I gave a good way to browse code." Look through your code and navigate and run into surprises that you didn't expect [while] browsing, like browsing a library. So I didn't—you know, nobody asked me to do it and I didn't know exactly where I was going, but I just started playing with it, this idea of browsing, and what I would like to do, so one thing I came up with was the idea that there would be panes and if you'd see something that—a heading that was of interest to you, you'd click it and then, in another pane, you'd see the expansion of that thing, whatever, and I tried to find a general—as general a thing as possible, and that turned into several different functions. One was just browsing source code, class, method, and then a window, big window, where you could see the method that you selected and edit it and—but also in debugging, there would be a pane that had the stack—every stack frame, and if you click "stack frame," it would show you all the variables at that level of the stack and you could evaluate expressions in the context of that level of the stack. And then there was the variable inspection window, where you had a list of variables and then to click a variable and it would show you its value and it'd let you change the value, so—and then there was one other fourth function, which was profiling, which was—used to be done by having—well, still done having an interrupt every 60th of a second or whatever [the] computer lets you do, and then it would

tell you what function you're in and what you're running, say what method in Smalltalk you're running, and count how many times you—that you sampled that it was this method versus that method, and so on, and the profile—profiler—and this was—profiles existed already at that point in that way, so I thought, "Well, we'll have to do better than that," and so I came up with the idea of, for every place you get interrupted, save the entire stack to get you there, so not only—it may be that you're spending a lot of time in the string concatenation routine, but why? You could've gotten there through any of 20 different ways and by looking at the stack, you see what ways you actually did get there, and if it turns out there's multiple ways of getting there, you'll see two separate pieces of profile.[4]

Hsu: Wow.

Tesler: Anyway, the other ones, browsing code—browsing source code, browsing the stack at a breakpoint or crash, and inspecting variables was what I called the Smalltalk browser, and Diana agreed, "Yes, you are browsing now. You're browsing source code. You're browsing stacks. Yes, now there's a way to browse." But that whole thing took—I think the first version was working in two weeks and then...

Hsu: Wow.

Tesler: ...it was—I think everything I told you was there in four to five weeks or so.

Hsu: Wow.

Tesler: And Smalltalk was—if you were fluent in Smalltalk, that's very typical. That's what people would do.

Hsu: Wow.

Tesler: And at that point, I'd had seven years of immersion or, well, five years maybe. I was there seven years.

Hsu: Right, so that was—what year was that at that point?

Tesler: The Smalltalk browser?

[4] Larry Tesler adds: And today that is called a "dynamic call graph profiler" and is available in Smalltalk derivatives as well as other IDE's such as Eclipse. I think that this type of profiler didn't exist before I put it in Smalltalk. In Smalltalk and some other systems, a profiler is called a "spy".

Hsu: Yeah, five years? If somebody—you started in ’73, that would be ’78, actually.

Tesler: I think the browser was around—yeah, it was around ’77, ’78. I can look in my files and see, actually, if you want to try to put this on a timeline.

Hsu: Yeah, that would be helpful. Wow, so yeah, it’s like all these—it seems like every subsequent graphical IDE uses at least one or more of those ideas that were in that Smalltalk browser.

Tesler: Mmhmm. One thing Dan added immediately was that there were just too many classes to browse—to scroll through. It took a long time to scroll through a hundred of them or something, and so he got the idea of having categories of classes, and at first it bothered me. I said, “What does a category do?” He said, “They won’t do anything, just—they’ll just—it’s just a way to group things so we won’t have so much to scroll through—completely—”

<laughter>

Tesler: That's the way you reduce the scrolling: just add another level gratuitously <chuckles> and the compiler recognized, you know, “in category blah” and when you say, you know, when you put a new method in. In fact, since it was a browser, he’d—you would go to the category before you create the class and that class would become instantiated in that category. And then there was a way to make a new category if you’re at the top level or that kind of thing.

Hsu: Okay, so a category was itself implemented as a Smalltalk class.

Tesler: It was kind of a wrapper to group together related classes for the purpose of scrolling reduction and...

Brock: Like an index, if you will, entries in an index.

Tesler: Well, it was kind of like tags these days. It was like a tag and with the limitation, you could have only one tag per class.

Hsu: Oh, right.

Tesler: But you could have a lot of classes with the same tag.

Hsu: Right.

Brock: Is there a throughline from these—from the Smalltalk browser work to...

Tesler: Eclipse?

Brock: ...features that people would be more familiar with, more casual computer users, in terms of Finder windows and—on the Apple—

Tesler: Oh, well, on the Mac...

Brock: Yeah.

Tesler: ...on the Mac, there're—and on Windows too now, there is paned window functionality on the desktop. On Windows, it's called Explorer, Windows Explorer, and on the Mac it's that multicolumn mode, where—or multicolumn view, where you have successive drilldown, basically. So it's an option if you want to use it.

Brock: So that's a direct appropriation of this Smalltalk browser work, that metaphor?

Tesler: I think that the guys that added those—that feature to those two operating systems, both cases, were probably inspired by either Smalltalk browser or something that was derived from it. I don't think—it could be just coincidental reinvention. They're not that identical. But that metaphor of clicking in pane three to get a list of things in pane four is—was the idea. I'm not sure it hadn't been done before, but it hadn't been done before in a GUI, which is—which was easy to say, because there weren't any other GUIs.

<general laughter>

Tesler: But there might've been—you know, I wouldn't be surprised if there were database type applications at IBM that had something like this on a VT-100 type of screen, whatever the IBM ones were called. But, you know, there was a lot of—some of this stuff is intuitive and will be reinvented many times.

Brock: Would it be okay if I asked a couple...

Hsu: Yeah, go ahead.

Brock: ..more sort of naïve questions...

Hsu: Yeah, yeah, sure.

Brock: ...of this sort at this point and then maybe we'll check in on...

<crew talk>

Brock: To return to a couple questions about Smalltalk, there's this—a way in which people talk about Smalltalk as a language, but then there's this other very connected aspect to it that it's sort of an environment in which one is using the language, so I wondered if you could help the naïve viewer such as myself really get a handle on this connection in Smalltalk between the Smalltalk environment and Smalltalk as language, a programming language.

Tesler: Yes, there's the Smalltalk environment—well, if you look at the books, the yellow book and the blue book and all that, there's one called The Smalltalk Environment, I think, right?[5] And kind of the same in the Byte magazine issue with the balloon.[6] And yeah, Alan did that purposely. Just one name covers both because they're so intertwined. There's a user interface. There's a programmer interface. There's a programming language, which [a] subset of it anyway, you could teach the kids. And I guess he likes the words—liked the term Smalltalk so much. It's a pretty likeable term, a little humor in the name. So yeah, so the Smalltalk language implements—is used to implement the Smalltalk environment, which is used to implement Smalltalk programs and I never—it never occurred to me that it was redundant, but it is, I guess.

Brock: Thank you, and I wondered if—in reading about the story of Xerox PARC, there is—there's often characterizations of these different groups, one might even say different factions within the laboratory or laboratories, and I was wondering if the community of people most engaged with Smalltalk formed. Was that a distinctive tribe within the landscape of other tribes at PARC? Was that a group that everybody recognized membership in? Were you a Smalltalk person versus another type of person? I was just wondering about that kind of local cultural landscape and how the Smalltalk fit in.

Tesler: Well, the people in the Smalltalk group, the Learning Research Group—all used Smalltalk. That was easy, except when Dan Ingalls went down into the microcode, but there were other people who were not officially members of the group, like I was for a while. Peter Deutsch wrote some Smalltalk code and...

[5] Larry Tesler adds: The Orange Book was titled *Smalltalk-80: The Interactive Programming Environment*.
[6] Larry Tesler adds: That issue of *Byte* was dated August 1981.

<Pause in thought>

Tesler: ...trying to remember who else. There were a few others. A few days ago, I ran into something about a couple of them, but certainly anybody was welcome to use it and people who were there for years probably at some point learned Smalltalk just so they would have had that experience and maybe they'd expect to learn more by doing it than just by attending lectures about what we did in Smalltalk.

<Pause in thought>

Tesler: Does that help a little?

Brock: It does, thanks.

Tesler: But I think, yes, that there was a kind of a community of practice in the sense that people who were not in the Learning Resource Group were affiliated with it. They felt like they were part of a community of users, but it wasn't a huge number, I don't think.

Brock: OK. Would you say it was in the mainstream of things in the laboratory or was it not?

Tesler: It was supposed to not be.

Brock: <laughs>

Tesler: Supposed to not be mainstream. It was a way to get something for Alan Kay to do to shake the rest of us up and be—whoever selected Alan to go there had to have known that it was gonna be—he was gonna be disruptive person and create—I wouldn't call it a faction, but in some sense it was a faction in that there were maybe a few zealots in the group who started out kind of idealizing Smalltalk and—but people matured and...

Brock: But it was there to provide some creative tension almost by design, if you will?

Tesler: I think the management chose Alan because they wanted to make sure that it wasn't gonna all be about xerography. It could've easily turned into, "This is a copy company and we're gonna—our goal is to find out ways this copier company can leverage digital technology," and that was the major theme, but one thing that all those people in the management knew was that you need to stir things up a little bit, and

have other things going on that you can learn from that are—that seem to be in a different area, but will cause you to free associate and figure out a way that this could be applied in your world.

Brock: Interesting. I guess the last of these questions of this type that I had was just, if you were forced to answer the question of what you believe are the most influential features of the Smalltalk work of this—of the 1970s, what would be the things that you would point to as the most important?

Tesler: Of all the things that that group did or all about—if it was just Smalltalk related?

Brock: Maybe Smalltalk in particular.

Tesler: Well, for example, the Dynabook concept is different degrees reflected in clamshell and tablet form factors and Alan had those ideas before he even came to PARC, but that's where he worked them out in any kind of detail. Dynamic object-oriented programming is maybe not as important now as it was during the GUI-centered period, but it's still important, and—[let's] see.

<Pause in thought>

Tesler: A lot of aspects of the UI started in Smalltalk, for example, overlapping windows. Of course, maybe Les Earnest had something similar sooner, but I am not familiar with it, but just in terms of what influenced other things, one thing that some of us did was intentionally go places where we could help influence that place to adopt something that we had before we got there. I would've been—it would've taken a lot of proof to convince me that we should not use cut and paste in the—on the Apple Lisa. There wasn't a lot of argument about it, but there was always—there's always pushback from somebody and—now, cut and paste didn't technically come out of Alan Kay's group, but that was how it got popular: because it was the way you edited in Smalltalk, and—but also, it—let's see. Well, I'm wandering away from the question. Yeah, so overlapping windows...

<Pause in thought>

Tesler: ...popup menus...

<Pause in thought>

Tesler: ...the IDE.

<Pause in thought>

Tesler: Those are the biggest ones. There's others. There's the model-view-controller framework. It's not universal, but—and...

<Pause in thought>

Tesler: Yeah.

Brock: Thank you.

Tesler: Stop there.

Hsu: I was wondering—you mentioned that you had worked on a lot of applications that were written in Smalltalk, and so, like, beyond the IDE, could you give us some more examples of some of those programs that you wrote?

Tesler: Let's see, what did I do?

<Pause in thought>

Tesler: Well, the early text editors. There was one called Mini Mouse and it was very primitive Smalltalk—Smalltalk-72, and not even running on an Alto, running on a Nova. But it was also called the Intuitive Typewriter. It was—felt like you were typing on a typewriter and I implemented it all, but Dan Ingalls helped with the design and it had a very clever thing in it. If you move the cursor somewhere and start typing, it would have to—the question is, are you over—'cause on a real typewriter, you're overtyping whatever's there before. It's not shifting it out of the way. It's not reflowing, and so people expected that to happen in those days. They expected overtyping, and so we kinda had to offer it—we thought we had to offer it as an option, but I didn't want a special mode that was, you know, overtype versus insert, and so Dan Swinehart came up with this idea, which was that we'll time it and if you—oh, I remember the real issue, sorry. The real issue that was hard to solve was, if you hit—if you used space the way it is on a typewriter, where the spacebar advances the print wheel or something by one space, it—if you do that when you've selected existing text, then the space makes—does nothing. It just leaves the character behind. The character you were over when you hit the space bar remains. The alternative would be that it erases it, because it's a space. It changes it to a space and the thing he came up with was that if you're typing at a certain rate of speed, we assume that you're giving us totally new text, and so the space will erase the character underneath, but if you're typing very slowly, then you can see that

you're about to erase that character. We're assume—we'll assume that it is just gonna move over, and—anyway, whatever. The—I'm not even sure why I went there, but it wasn't really what you were trying to find out anyway.

Brock: It's just an example of what you could do...

Tesler: Oh, what I was...

Brock: ...with Smalltalk.

Tesler: This is—right, so the thing is, that—this is what I implemented. I implemented that with a bunch of other heuristics to try to decide how to map the standard functions of a real typewriter to the simulated typewriter that we were using, to see whether we could come up with the best of both worlds where, if you were—the thing we didn't realize was that if you're—after half an hour, you never want it to behave like a typewriter again.

Hsu: <laughs>

Tesler: But the way we found that out was by implementing it and testing it and that's what people said. They said, "Oh, that's what I expected to happen, but I'm glad it didn't." Let's see, other things I did in Smalltalk. Mostly, it would be implementing system classes for other people to use, and...

<Pause in thought>

Tesler: ...text editing… The Smalltalk decompiler, implemented that, but that's for programmers.

END OF THE INTERVIEW

