Oral History of Adele Goldberg

Interviewed by: John Mashey

Recorded: May 10, 2010
Mountain View, California

CHM Reference number: X5823.2010

John Mashey: Okay, so Adele, let's start all the way back. You're from Chicago originally, is that correct?

Adele Goldberg: No, I was born in Cleveland, Ohio.

Mashey: Oh, okay, oh.

Goldberg: We moved to Chicago when I was eleven and then I was in Chicago until I went to college. Then I went to the University of Michigan in Ann Arbor. And then I went back to Chicago for graduate school at the University of Chicago.

Mashey: What were you studying as an undergraduate?

Goldberg: So I studied mathematics and I was actually in Ann Arbor for a total of three and a half years. The first half of my senior year I left in the Spring to go to Europe. I was accepted at the university in Munich. It wasn't junior year abroad, but the people who were running the program were supposed to arrange housing and didn't. So when I got there I spent a of couple weeks looking for housing and I said, "You know forget it. I'm going to go enjoy Europe." So I went all over Europe and up to England and way north and ended up in Israel and then came back to the United States in the Fall. Everywhere I went there was an IBM building and I had been pondering what was I going to do with this math degree because originally I was doing it to teach high school math, which is what my mother was trained to do. I discovered at the University of Michigan to get a teaching certificate you had to do a public speaking class, and I wouldn't do it. It scared me, which is amazing. Well in those days I didn't think I had anything interesting to talk about. It wasn't until you do something that's interesting to talk about that you can talk; so I never got my certificate. So I was thinking what was I going to do? And I saw all those IBM buildings and I thought, "Wow, I can live anywhere in the world if I could learn to do something that is important to IBM." I had a cousin who was a statistician for them in New York and he explained a little more. And I had taken already a computer class. In those days we were punching cards.

Mashey: I remember them well.

Goldberg: I got it in my head that I should get my Bachelor's and then go get a Master's degree so I'd learn more about computers. I got a job at Michigan at the Center for Research on Learning and Teaching, working with Karl Zinn who is one of the pioneers in computer aided instruction. He hired me mostly because in those days I knew how to speak German and they were teaching German class using an IBM 1500. It was very straightforward programming in my mind. I saw programming as the grandest world for problem solving. It's just perfect for me. So I applied for graduate school and picked Chicago primarily to go home to see if I wanted to live in Chicago again. So it was sort of this "go get your Master's degree" and it was 1967.

And in 1967, I needed a fellowship. Women just were not going into those [math, engineering] fields. And women were not getting fellowships until the universities realized that, because they're businesses and they need students, they were going to be forced into bringing more women in because too many of the guys were getting drafted. Years ago, Thelma Estrin told me that she had the same experience in World War II, that she got this wonderful job at UCLA running a lab there because the guys were all off in the war. I said, "It's too bad it takes a war for us to get in." But I did very well and I was encouraged to take exams to go on for my Ph.D. They ask you when you take your exams, "Well what are you going to do your Ph.D. [dissertation] in?" And what I knew about was educational technology. That's what I learned at the end in Michigan and it was still something interesting to me. Chicago had mostly Atomic Energy Commission funding. These were the years [at the University of Chicago] of Bob Fabry and Maurice Wilkes, who used to visit, and they were building a machine they called the Magic Number Machine, which was capability based hardware [Fabry, Robert S. "Capability based addressing". ACM *Communications 17*, 7 (July 1974), 403-412.] . Chicago didn't really have funding for educational technology. But one of the faculty, Roman Weil (who is still there in the business school) asked Patrick Suppes who was visiting [from Stanford University] whether I could come visit him. Patrick Suppes was one of the two directors of the Institute for Mathematical Studies in the Social Sciences [IMSSS] and he said sure. So I was at Chicago for two years, got my Master's, and packed up to go to Stanford, not as a student but as a visiting student, still a Chicago student. After the first year, Chicago accepted a dissertation proposal with no strings attached. You could be anywhere. You just had five years to get it done. Stanford had offered me a research associate position so I didn't leave California.

Mashey: So much for Chicago.

Goldberg: So much for letting a student go away. Well the alternative was going back to Chicago and continuing to teach programming which would have [actually been fun]— ordinarily teaching programming class would be fun and we did it in an interesting way. We did it as comparative programming languages so the one course I taught [actually team taught with Chris Brown] when I was there was SNOBOL, PL/I, and BAL (assembly language). And the way we taught it was to teach you how to build the elements of one language in the other so that in the end you could see what the language was designed for and then what it meant to write a language [in another language]. Unfortunately, it was 1968 Chicago and those were not happy years at the University, nor in the City, because that was the year of the Democratic Convention and there was a big sit-in [at the University]. My students were sitting in at the admin building and I learned very early about the politics of universities, trying to protect students from themselves because, of course, at the University of Chicago, if they [the students] don't come to class they are supposed to be flunked. So I negotiated to let them make up the course by a certain date and then they would be okay because otherwise, if you flunk a class in Chicago, you might as well go home. So it was an interesting experience.

Mashey: Interesting time, yeah.

Goldberg: Yeah it was interesting and I'm not sure if that turned me off from working in the university. I know that when I finished my degree at Stanford I did teach for a short time visiting down in Rio de Janeiro [at the Pontificia Universidade Catolica do Rio de Janeiro], but I think it wasn't really a commercial versus university tradeoff. It was more: Alan Kay wants me to come play?

Mashey: Oh yeah, sure.

Goldberg: I'm going.

Mashey: So how did that hook-up happen?

Goldberg: IMSSS, the [Stanford] Institute, had both people working in mathematics and in reading education. Richard Atkinson headed up the reading part. He later became the chancellor of the University of California. There were a lot of graduate students doing some very interesting and novel work but on the big timeshare systems. [John] Dexter Fletcher was one of them and he said, "We need to get involved in the ACM and we need to get involved in the community around us interested in computer-based education. We can't just be at the university." So he and I did two things. We went to an ACM meeting down in Anaheim [and got involved in SIGCUE, the ACM Special Interest Group on Computer Uses in Education], and we also got involved with more of a local user group that was looking at computer uses for kids. That means you're not so insular. You've got to find ways not to just be in your own building. Because of our community work, I met a number of people who were either going to or were already at [Xerox] PARC. John Shoch was one of them and so I think that's how Alan became aware of what we were doing at Stanford. Besides teaching [elementary level math] courses, I built a [college level] system in which students could study the predicate logic by essentially inventing their own axiomatic system, really so they'd know how hard it was. [The online theorem proving approach incorporated in the online system was] a way of programming. We were also teaching Logo, the Seymour Papert work, and there was more of an assembly language level language—I think it was called SLOGO—that we were working on with kids, just trying to understand different forms of bringing computing into the classroom. So Alan became aware of our work. He had already come to the conclusion that to understand the software for the Dynabook you couldn't just start with adults. You're going to have to bootstrap yourself. You're going to have to start with the kids and see if that [what you learned, created] applied to [or was useful for] the adults.

Mashey: Okay, so that got you over to PARC. When was that done?

Goldberg: So that would have been the summer of 1973 as I had spent most of 1972 in Brazil. I came back, finished my degree, and went to PARC at that time. I went to PARC pregnant with my first daughter and so those beanbags were not as attractive to me as they were to other people. Alan always said the purpose of the beanbag was so that you would sink into the beanbag and it would be hard to jump up and attack somebody when they're giving a talk.

Mashey: Oh, great.

Goldberg: Which was funny. What you don't understand is once you sunk in and you were pregnant you couldn't jump up. Other people had to bring you up. That was pretty funny. I remember my folks visiting and taking them there and they just were tsk-tsking. "This isn't a workplace. This is a playground" which it was and that's a good thing.

Mashey: So talk about those first few years then in terms of what you got to play with.

Goldberg: Well I remember when I got there thinking "This is too good to be true" to be given such free license from a large corporation. By this time I had spent a couple summers working for IBM while I was in school and I had a very clear sense of large corporations and how they were organized and how they ran things. I didn’t see Xerox as any different so I was impressed with the forward thinking that gave such free hand to so many people, so many smart people. But I also felt that it wasn’t going to last because when you start with nothing and you ask for a budget and with that budget you build something quite novel, it’s going to be hard to convince people that you’ve learned everything there is to learn with that novel piece of hardware and you need to recapitalize. It actually happened faster than I expected, that we started getting more of the tension of resource tradeoffs. But the first couple years we didn’t feel that way.

[At the time of my joining PARC,] Smalltalk-72 had been designed and the challenge given to me was “Do you think children can learn this language? How would you approach it?” Again in those days the major influence, I mean there was influence from what I would call the education world: Benjamin Bloom who I knew at the University of Chicago, and Jerome Bruner was somebody whose work and approach to curriculum was very admired. But the biggest influence was Seymour Papert and his group and they didn’t believe in curriculum. They believed that if you gave somebody a toy that intrigued them they would learn something from it. I think the fallacy in that is that they actually expected their students to learn something fundamental about mathematics and they made such claims [that the young kids were inventing mathematics], but there wasn’t any transferrable guidance— if they were successful it would have been by the power of personality [Papert's mostly] and that doesn’t transfer. If you’re going to have broader impact in the school systems, you’re going to have to give better guidelines and you’re going to have to tie what you’re doing to expectation, more so now with accountability rules than back then. But still the question was always in front of you: what will the kids learn if they do this? In the early ‘70s, programmable little calculators were just in the schools. Computers were not something everybody had their hands on. You couldn’t argue the importance of just learning about what a computer is. It had to tie to other curriculums.

So my challenge was to balance the exploratory, what we would call more of an inquiry approach (which is how we think about it at the San Francisco Exploratorium where I’m [currently] involved) with at least a curriculum guideline. As it turned out, we were quite successful, not so much because I had a genius idea or anything but because the nature of the language gave us the answer, which is, “If I give you something that you can play with and extend, even a piece of paper with a paragraph and I say it’s not written well, rewrite it, that’s easier than giving you nothing and say make something; you know, giving a blank sheet of paper and starting to write.” So the lovely part that has proven true for professional programmers as well as kids is when you start with something, an object that does something, and then you put many objects like those together and have them interact, and then you extend and make them behave a little differently, you can take a very incremental approach to learning how to control a computer system. Actually we took a systems approach. It wasn’t just lines and lines and lines of code the way many projects teaching programming using BASIC found themselves doing. You really were mapping to their natural understanding of what were the actors in the application they were trying to build, [and how do these actors interact]?

Mashey: So some of that early work I assume was done in Altos. Where did the Dynabook, old Dynabook idea come from? Talk some more about when you thought that would happen and what you wanted that to be.

Goldberg: So clearly I went to Xerox to join Alan Kay’s dream. Having spent years and years doing educational work on timesharing systems, having spent many years working with computers and tutoring in public schools including here in East Palo Alto, the Ravenswood School District, I was not comfortable with [shared, invisible timesharing] [which propagates the idea that, if] you're going to learn, you go to a place to learn. What I believed is, is that you have to bring your tools of learning with you wherever you are. Obviously that starts with your brain, but if computers were going to be successful in changing how we learn and we were going to deal with lifetime learning, then they [the computers] had to be with you [be able to go where you go]. In walks this wonderful man with this cardboard mockup of a carry-it-with-you computer, which didn't look so strange. [Schools had programmable calculators by that time]. But in the schools, calculators were tethered. They were locked to the desk, right? You didn't take them home. You didn't carry them with you. They cost too much money. But we could see where it was heading eventually. So when I first learned about the Dynabook dream, it just mapped exactly into what was bothering me about the timeshare work. Alan had done work as his Ph.D. dissertation at the University of Utah, of course, on the FLEX machine, so he had been at it for many, many years trying to figure out how to create this computation and communication, multimedia device, and had been mocking up a whole series of them over the years. I think there was one called the KiddiKomp. But when I stepped in he had already had this Dynabook mockup. He'd go around giving talks and showing this computer and I think what people don't realize is, when you give a talk with the enthusiasm with which he gave it, people thought he was marketing something he had.

Mashey: A product he had, yeah, yeah.

Goldberg: And so we'd get letters to buy them. Sometimes you could hardly dissuade people that it wasn't real because we had it simulated on the Alto. For people who understood what they were listening to, it was very intriguing. But many people, as we've talked about in the past, many people just thought he was crazy and we were crazy and that it would never happen. It could never happen. Nothing motivates a research team more than to be told you can't do it. In fact, it's a way in which you get your researchers to stop thinking about what's hard and just do it because it's like telling somebody in sports "You can't run that fast." "Well, wait, I'll show you. I can do that" Well, it's obviously the same motivator for software people, software and hardware people.

We took the Altos to the kids. When I started at Xerox we were in a building up on Coyote Hill, not the current building, and there was a basement area not being used. I think the POLOS, the [PARC On-Line] office [system] group, were down there but there was space. So we carpeted it and we put our group's Altos down there (there was kind of an allotment of Altos to different groups). And then we brought students. We negotiated with the [local] school system for the MGM students, mentally gifted minor students, to participate in this experience. Interestingly, one of those student's parents lives on my street where I live now in Palo Alto and they said that that was the primary influence for him to go into computing later, which I hope he wasn't shocked to find out in those days that [available computers] wasn't what it was like [at PARC]. I'm sure he was. We also arranged it so one of the students who wanted to be the teacher could have her own class, and so she taught a class (Miriam). These kids were having a great time because some of them loved to paint. Some of them loved to dream up ideas. If anything, we learned not everyone had to program, but that [group programming] was a great way to teach kids how to collaborate with one another, that everyone could have a different role. We didn't use the word "team". We didn't use the word "collaboration". But they almost naturally fell into the role of saying, "Well I don't really know how to do it but you do. Here's what I want to see." We did a lot of that.

And on weekend Saturdays we'd have classes for Xerox employees' kids and that was a great way for me to meet the people in the other laboratories because, of course, we had two computer laboratories but we also had three others, chemists and physicists. Their kids came for class and that was fun and that allowed them to understand more about what we were doing because otherwise you sort of wonder what's the point? What are they spending money on? What's going on? It didn't look like hardcore science, which it wasn't.

Mashey: So what years was that going on?

Goldberg: Whoa, 1974, I'm going to date this… for several years. The way I'm dating this you'll laugh because we ended up creating a resource center at Jordan Junior High School and taking the Altos there which was a bit of a bureaucratic struggle but I'll leave that story for some other day. I remember that I had a one-year-old and I took her with me and we had snow in Palo Alto and the junior high school kids were having a snowball fight and that was like in the winter of '74-'75 so from '74 to '76. Smalltalk-72 is what we were teaching at the time; it was a very difficult language to teach. I understand the motivation behind it but basically it had the core idea of objects that were sent messages, but an object could decide to gobble up the message stream. So depending on each token in the message stream [and which object received the initial message], something different could happen, which meant that you really had to mentally think through the implementation to know what was being said in the code. It wasn't easy to read the code. Out of that experience I remember begging that we don't allow that because you could gobble it up and evaluate, gobble it up and take it as is. I mean there were all different ways of gobbling it up and it depended on what the implementation of the message was to understand the gobbling. Adult professionals have problems with that kind of language. It was beautiful because it was very iconic and there was a lot of research going on at the time on iconic programming. David Canfield Smith's Pygmalion was one of the first languages for just expressing what you want to have happen with imagery, although there were not good solutions to the complexity of insufficient screen space.

Mashey: Yeah, right. So then I guess maybe talk about sort of the changes that went on in Smalltalk deriving from that, the different versions.

Goldberg: So there was a Smalltalk-72, which had that characteristic [message gobbling, mentioned earlier], very small system and the graphics was very oriented to line drawing. It was the turtle metaphor from Logo. We had Myrtle the Turtle and that was a little icon and so that was fun for the kids for learning to draw but limiting, very limiting. So in parallel to language design there was a lot of work on painting systems and animation systems and music systems to be integrated so you kind of did them in parallel and figured out how to merge. '74 was a new implementation. We had started out on Data General Novas programming in BCPL [the Alto initially emulated BCPL]. [The Smalltalk virtual machine was defined in terms of byte codes and] the Smalltalk byte codes would then be interpreted. So there was a new implementation in '74 and then in '76 there was a new language which essentially did away with this message gobbling, but still had the icons.

And it was in Smalltalk-76 that Dan Ingalls did the first *bitBLT* which is the block transfer of bits, the underlying system code for being able to manipulate the bits on a bitmap display. That bitmap display was supposed to be a simulation of what we thought would happen in a flat panel display but it took on a life of its own. Things like that happen. I always say to people when they're fretting about some project they're going to do or some new job they're going to do, "Quit worrying about failure. Failure's easy. Worry about if you're successful because then you have to deal with it." [So in some sense the success of the bitmap display took away, at the time, interest in inventing the Dynabook's preferred flat panel technology.] Smalltalk-76 is where there was some very beautiful examples of galley editing. Diana Merry did some wonderful implementations of a galley editor that mixed the text and pictures. I remember we were going to have a big demo on a Dorado with some corporate people, so we wrote a text that described essentially how paper flowed through a copier duplicator. We were starting to get the idea that we better figure out how to explain what we're doing less in educational terms and more in how it would benefit the corporation, something I think would have been good to start with, but hindsight is so smart! I did an animated sequence of the paper flowing so you could actually see the still frame all labeled so that it looked like a sequence of images. And we just shocked them when we hit a "run button". I think the one we surprised the most was a PARC guy, Bob Spinrad. He was quite surprised by it [the embedded animation with its own editor] and that felt good to have a little surprise.

Mashey: You mentioned Dorados in there. Why don't you talk about that a little bit. When did that come in?

Goldberg: So basically after the Alto there was a series of hardware designs done in the computer science lab. Alan's group, with me in it, were in the Systems Science Lab [SSL] and our manager was Bert Sutherland. Then in the Computer Science Lab [CSL] the initial manager was Jerry Elkind with Bob Taylor as sort of associate head. I'm not sure what his title was exactly. Frankly given his history at ARPA and funding at ARPA I think everyone deferred to him more as the head and Jerry more as the liaison with corporate. Jerry had come from Bolt, Beranek and Newman. CSL built first a Dolphin and then the Dorado and then the Dandelion which became the hardware system for the Star Workstation which was announced in 1981, so that's a lot of machines being designed from 1972 up, in a short amount of time. Because we weren't getting recapitalized too easily, one of the things we discovered is you could take all your Altos and sell them to somebody else within Xerox. You didn't get real money. You got credits and with that you could get the next machines. And in that same timeframe, because other divisions were getting the machines, Xerox Special Information Systems Group down in Pasadena had gotten Dolphins as well and were experimenting with Smalltalk. I didn't find that out until more like 1980, or 1979, and it plays a significant part of the future of how things played out. [Subsequently, Xerox formed an AI business headed up by Whit Haney to sell Interlisp-D systems on the Dolphin.]

The other hardware system that was a follow through was Notetaker. So this is where we collaborated with Doug Fairbairn and designed, you could call it, a portable computer. I called it a "luggable". It wasn't physically that different in terms of size than the Adam Osborne machines. You could carry it with you, lug, lug, put it under the seat in coach. I understand that you couldn't do it in first class. It was apparently too tight. Larry Tesler did that [traveling with the Notetaker on an airplane trip]. It had batteries. It had dual 68000 procesors in it, one of them managing the bitmap display. What we were trying to do was move closer and closer to the Dynabook ideal while the other machines were getting bigger and bigger. I mean a Dorado—you needed air-conditioning for that ECL machine.

I remember in 1977—the Smalltalk-76 system really pushed more to have a class hierarchy so each object was described in terms of a class of similar objects. Then you created instances of the class. Those were really the active objects. But we were aiming towards a system where everything was an object, which meant that a class would be an object too and you could send messages to the class which would handle mostly information and actions shared across all instances. So there would be a class hierarchy where you would say, “Oh, I’m a traveling vehicle just like that one except I’m different in this following way and I can be different in how I implement something or I can have more properties and understand other things.” The typical business example would be: there are bank accounts but there are special bank accounts. So we have this whole thing laid out and it was very compelling because it fit right into the curriculum ideas for how to teach kids. Give them a starting point and really help them learn how to refine it. We were really exploring this notion of "programming by refinement" and really seeing a whole new software engineering process where you programmed iteratively. You started out with a prototype and you kept changing it and working it until it was what you wanted, and then you started figuring out where performance could be improved.

And in 1977, there was a great need to teach the executives, the top ten executives of Xerox, about what we were doing at PARC. This [education effort] was an initiative started by Bob Taylor I believe or someone in the Computer Science Lab, where the chairman and president and the top ten guys were invited to spend a couple days at PARC to learn. The basic idea of what they should learn is about "modularity". Well I think the powerful idea they were trying to be taught is that a piece of hardware changes depending on the software that’s loaded into it. It’s a very obvious idea today but not in those days! [And that new software could be created by composing otherwise independent applications or modules.] The combination hardware/software, that was the package; that was the product; instead of saying, “We built this general purpose piece of hardware and it can be whatever you want it to be.” [I was asked to create an environment or simulation in which these executives could have hands-on experience with the idea of customizing software. The simulation kit we created met the goal very well.] They [CSL] had not been very successful I think in getting enthusiasm for text editing systems. The chairman at the time, Peter McCollough, had come to PARC for a big demo of the Bravo text editor that Charles Simonyi had done. When Bob Flegal, who worked for me, saw McCollough the next week at corporate because he was there for some social service project, asked, “Well what did you think?” Then McCollough said, “I don’t think I’ve ever seen a man type so fast.” He missed the point. This particular problem went on and on and on, where we didn’t know enough about how our customers, our internal customers, thought and they didn’t have a particular mission for us that they could focus their questions around; there was no business plan [or focused research vision expectation by the business managers]. It was hard. I mean it was just hard.

Mashey: How did PARC get funded in the first place? It’s pretty far away from corporate headquarters.

Goldberg: Yeah. I think the instigation was Jack Goldman. Dr. Goldman was the chief scientist and I believe he made the case for a computer industry coming in. The Xerox vision, which was actually then and now a perfectly reasonable one, which is taking information from one form and turning it into another form, basically the copier/duplicator distribution model where the original could be paper or it could be electronic, you know the office of the future. Jack had that idea. He hired George Pake out of Washington University, St. Louis, a renowned physicist. They wanted to be near a university and they explored lots of them and picked Stanford at a time when government funding was low and when quite a number of respectable, fabulous researchers who were used to getting ARPA money were no longer able to get the money. These researchers were coming out of Berkeley, coming out of Stanford, and other places. They hired Bob Taylor who knew them all from the ARPA community.

But they didn't start a culture of teaching all those people what Xerox was all about. What they said, which wasn't unreasonable but wasn't good for the company [in the long run], what they said was, "Work on anything that you think in five years could have an impact on the company". And they did, and they were successful and the Ethernet, color copiers, high end printers, all made a big difference in the company. The problem is that the thing that everyone remembers the most was a small computer [the Alto] and the company wasn't prepared to be in the [small] computing business. They did buy a big computer company, SDS [Scientific Data Systems], but no one at PARC was involved in that or wanted to be involved in that purchase. I don't know if that was an issue because I wasn't involved in it. But I know in 1977, when we taught the class and they asked Alan's group to do the hands-on laboratory so I designed a customizable simulation system [we called it a "kit"] where they could create models of different copier/duplicator centers, putting in the [scheduling and service algorithms] and data themselves, and everyone could create something different. And they could look at each other's machine and see something different going on. It was at that class where I found out the president of Xerox had been at IBM and knew how to program, so a little different than we knew [before that experience]. I think ultimately it went over very well but they still didn't have a business plan [or interaction plan for research to communicate with the business units]. In 1978, when the Notetaker was designed (again it was Smalltalk-76 running on the smaller machine), we had the idea that "Ah ha, this is perfect because by now we know two problems that Xerox has that we actually have a solution for" [document handling for technical support representatives and on-demand publishing].

Alan had gone on sabbatical and I was acting manager and then he didn't come back and I became the manager. I remember going back East and meeting the people who were responsible for Xerox's manuals. These guys were amazing in what they were doing because they drove the manual design from an inventory database and then they used mechanical translation to translate into multiple languages. They essentially had negotiated what Spanish to use that would be useful in all South America, and what French to use that could be accepted in Canada as well as France. They had a technique for idiomatic expressions that allowed them to do the translations efficiently. And so we said, "What problem do the tech support guys have?" At the time, Xerox was getting a quality problem with the customers. A tech support guy would go in to fix a copier and he wouldn't know how. He needed to look something up in the manual and the only way he could do it was to go out to his car, open his trunk and read this huge set of books that weren't up to date, even though they [the writers of the manual] had the [parts] database up-to-date. They knew. We said, "You could dynamically do that. We can put all those books on a little computer. He can carry it in. He can lie on the floor looking at stuff and looking at the diagrams on our little computer. We have this bitmap display." I'm afraid they thought we were crazy. We even said, "Maybe someone could help us and we'd hook it up to the duplicator"— by this time the duplicators had little Ethernets inside of them. We could just hook up and have a diagnostic tool. Either we didn't express ourselves very well or they just didn't understand. It was a very big disappointment.

Mashey: Well, for what it's worth, we had some similar experiences at Bell Labs about that same time with the same piece of the world in terms of maintenance.

Goldberg: So that was one of our runs at working more closely with the company. At the same time that Larry Tesler and I worked on this tech support idea, we were invited onto a planning group of the Xerox Publishing Group. Larry recommended on-demand publishing, and this was a group that was really quite innovative because they were looking to publish videos and they were doing all the sales support training videos. I remember taking Xerox sales training at Leesburg. I mean we got ourselves involved. We said, "We have to be more involved." They too didn't think that on-demand publishing was a very good idea and it's taken until now for the idea to be talked about. It's taken a long time. We were always so far ahead of the market. We needed to figure out how to get to the market. So we had Smalltalk-76. It still had unusual characters. We had given lots of demos. We were giving talks and XSIS [Xerox Special Information Systems] in Pasadena, had brought the CIA in and it turned out that they were doing a project with them to invent a whole new workstation approach for analysts, and they were doing it all in Smalltalk. I remember going for a visit to understand what they were doing. It was a combination of people at Langley and at NPIC, which is the National Photographic Interpretation Center, and I remember meeting this gentleman who was by profession a photographer. He showed me this prototype that they had been working on and the first thing that floored me was I thought they were supposed to tell us when they took our research and started doing something with it. But the real thing that floored me was there were no books. There was no documentation. There were talks and in those talks we would say, "If you want to know how to do something, you just look and read the code and it pretty much self documents." I mean we were so full of ourselves [really, in our belief in what we were creating]. And I said to them, "Whoa, wait a minute." It was a different user interface. Ultimately, it was the very first spreadsheet where the spreadsheet was an object, where all the cells were objects, and therefore a cell could be a spreadsheet. In those days the market didn't have that. It was all hooked to a database and it was all getting CIA data coming down and it was a remarkable *tour de force*. And I said to him, "I don't understand how you did that." I mean I just couldn't believe this non professional. He said, "Well, I'll show you." And so I said, "Okay." "Hit Control-C" (which is how you then get into the code and interrupt it to see the code that was executing). I remember saying to him, "That is the ugliest piece of code I've ever seen." And he said, "Oh that's okay because it does what I want it to do and the professionals down at XSIS are going to rewrite it all. This is the spec." Suddenly the prototype became the spec and it didn't matter how. What mattered was what [functionality it demonstrated needed to be available]. Talk about learning from your customers! That was the beginning of my getting interested in having customers. Alan never wanted to have customers because you have to support them. If you support them, you're going to do what they need and not what your vision is. I'm sure to this day he is livid angry with me because I marched down the software engineering lane and got customers. But we weren't stuck. We just needed more ideas. We needed more participants.

Mashey: So then that's about the Smalltalk-80 time right?

Goldberg: Yes, it was almost Smalltalk-80 time. It was time for the next language and the question was, "How far would we push the uniformity?" I wanted to do more on non-proprietary Xerox processors. I wanted to understand if you're going to have a language of this type did you need special hardware? There was actually an assumption that you did and I just didn't feel that we would learn for sure unless we brought in outside computers. This is also the time of the Sun Microsystems start-up.

Mashey: Yeah, sure enough.

Goldberg: So Andy [Bechtelshein] had done the work at Stanford [and Sun Microsystems started]. In fact, I was one of the earliest purchasers and I got the money by selling off all of our machines, all of our Xerox machines. And I know that later it became a problem. I mean I wasn't a pariah but I know because Bill Spencer, who, by the time there was an issue, was in charge of PARC, would come and tell me "They're breathing down my neck again about my letting you buy those machines." I said, "But you know the answer. We can't just let the world go by us. We need to understand and we need to know if the software we're doing will run and run well on these machines, or whether we need to do special processors." Before this time, however, I got it into my head that we should do the next round of Smalltalk in a participatory manner, more like what you'd all know today as open source. Smalltalk was a flavor of system in which the code was always there, was open source in the sense that the system, as written in itself, was accessible to all programmers; but it wasn't being built by a community. Bert Sutherland, head of SSL, was my manager and supported our building that external community. I said, "I don't want us to do all this work without [Xerox Corporate] permission because I think this time we need to publish everything." Alan had his hand slapped when in the early '70s..., help me with the name, who did the Whole Earth catalog.

Mashey: Oh, Stewart Brand.

Goldberg: When Stewart Brand came into PARC and there was a book Cybernetics 2? [*II cybernetic frontiers*], half of it was about the "secret" company and Stewart had pictures of PARC and so there had been publicity that I guess hadn't been properly signed off on. And so there had been a bit of a "You don't get to publish for awhile" period. I think Alan took it too seriously. He didn't like having his hand slapped. We did start publishing again more in the '76 timeframe including the personal dynamic media paper that Alan and I did for IEEE *Computer Software*. I wanted to go to corporate and ask for permission and it was a little risky because you didn't know what would come back. There's no reason for them to say yes. Lucky for me, between the time I suggested it and actually asking, Xerox's venture group had bought a lot of equity in a little fledgling computer company that wasn't doing very well called Apple Computer and they had arranged for the Apple executives and then later the Lisa programming team, to come for a visit. I kind of parlayed that into, "Well you're willing to give it away. You're not interested." And I wrote this motivator about why we should be able to publish it all because we were planning to write a book and we were planning to have this community implementation event. I think it went to one of the principals at the venture group who then asked Jeff Rulifson his opinion. Jeff was on some special leave from PARC to corporate and he wrote "Oh sure, publish all this". So I had it [the permission] on paper.

Mashey: Safe.

Goldberg: And so then with Bert's help we enlisted Tektronix, Hewlett-Packard, DEC, and Apple. Intel wanted to do it [participate in the community creating Smalltalk-80] but the rule was you must have an internal software team that works with your hardware team because the question we were asking wasn't "Do you like the language?" The question we were asking is "Is there anything about hardware that would make for a better performance?" And then we worked with the folks at Berkeley [Dave Patterson and David Ungar, the SPARC team], and with Eliott Moss at UMass, and Ralph Johnson in Illinois, but basically the initial release then went out and we ended up with a spec for the implementation of what was Smalltalk-80. So, of course, we were forced into straight standard ASCII because not everyone was going to have control over their fonts. Now they would, but back then they didn't, so that we could have had all these special characters. I don't know that the special characters were all that important. They did give a fun flavor for the kids. We tried to write one book with the whole team and that didn't work but we had nice write ups on Steve Weyer's *Findit*, which was one of the very first efforts to explore [online books]— can you find things, browse and find things, information on an electronic book versus a hard book? He did that as his Ph.D. in the education school at Stanford. We had all the work going on with constraint-based programming, ThingLab with Alan Borning's thesis [for Stanford, Computer Science]. We had later the Alternate Reality Kit [by Randall Smith's Stanford Computer Science PhD]. We were still doing these educational systems, Laura Gould and Bill Finzer were doing the Rehearsal World where, if you want to understand what an object did, you had a stage and you rehearsed the objects and through the rehearsal essentially evolved the programming, but also constantly looking to see where the rehearsal world language wasn't strong enough, where just adding objects to the library wasn't going to be strong enough. The system had these trap doors into C programming or lower level programming. We tried to write, I think, too much, into the one book.

Finally, what we decided to do was that Dave Robson and I would write and we would split it up into three books, four books actually, but only three came out; a language book that Dave and I would do; a user interface book that no one wanted to do so I did because it was more like a user manual; and a book on implementation, the experience of everybody that was edited by Glenn Krasner. And then there was going to be an applications level book which was basically the model-view-controller metaphor. In order to get the language book done we kind of organized ourselves so that Dan Ingalls and I [made the final decisions]. We'd have an aspect of the system we'd need to agree on like what would be the collection classes? How would you provide collections? I had run a group at PARC that did APL that I was fascinated by so we actually had a lot of the APL stuff [collection manipulation] in it. We'd go to the group. We'd have this plan. We'd have a big debate. Dan and I would make a decision. And then if you thought you weren't heard, you went to Dave as the ombudsman and he'd listen and if he thought your complaint had merit he'd come to me and Dan and he'd say, "Listen you guys, you are screwing up. You didn't listen." It was a very interesting organizational dynamic because Dave never came to us. So at some point I said, "This is going too well. Nobody's complaining. This is a group that has opinions, not a lot of them, but they have opinions. What's going on? What are you not telling us?" He said, "No, no one comes to me." I said, "Why not?" He says, "Because they know they'll never get through me. The goal is to get done." So this was like a very different model than our previous history where it was a smaller effort. This was a much bigger effort and it got us involved with the world in a really fun way. The book came out in 1983 [although between getting started in late 1979 and the book release, the August issue of *Byte Magazine* came out with all the featured articles about Smalltalk-80]. That's how long it took. It took a long time both as a seminal book in object oriented language, the structure of the book which I really feel good about the quality of that, and the fun doing the artwork. Bob Flegal and I did the artwork. I realize now when you want to be an artist but you'll never be an [professional] artist and it's your book, you get to do it!

Mashey: You can do what you want.

Goldberg: You do what you want. You get to do that. When it came out I think it took a lot of people by surprise and I think one of the reasons why we managed to do it was that Xerox had built the new building extension. [PARC] was a three layer building with pods and there were three pods, and they were going to add a fourth. They asked John Seely Brown's group to go to a different building. They essentially were pushing him out. I went and looked to see where they were pushing him to and I thought, "Oh this is a great building. Who wants to hang around in all that noise?" So I said, "Can I go too?" I was advised not to do that. It was politically not good but it meant we had quiet. We were left alone and we could get it done, plus at lunchtime we had a little band. We didn't play very well but there was no one there to complain so we got to play music! So we were able to get the system and book done and, of course, that was life changing for me. When we got it done, we had a language now that other people were going to use. Other people had opinions about, eventually went to standards work because it was commercialized. It was a great time to say, "Okay we've reached a plateau. We've been trying to solve this problem about personal computing. We made a contribution. That wasn't the only problem we were trying to solve and we turned to ask questions about the interpersonal computing. That was when I asked— by this time I was a research laboratory manager and I asked, “Could I build the rest of the lab up in Portland, Oregon” because by this time we knew all these people all around.

Mashey: Oh, sure all around.

Goldberg: ...who were already enmeshed in the culture. And so we built the very first 24-hour essentially virtual lab with video conferencing which spurred the CSCW work in collaborative systems which has still intrigued me to this day. I mean those problems aren’t solved yet but it was the transition. Then by 1986, we had the CIA as such an important customer. Xerox was not going to do anything to commercialize. They [the CIA] wanted to deploy. We had done implementations on 68000-based machines. We had a 68000-based plug-in [board] to Xerox personal computers [Xerox 820] that the guys down in the Dallas Development Center had done and they weren’t going to service it and I just thought “You can’t do this. You can’t do this to a research group.”

Mashey: Yeah, right.

Goldberg: You have to transfer it [to a commercial organization] or not [allow external use]. So we asked “Is somebody going to do something with this and, if not, here’s a business proposal for a spinout.” And I’m going to make it sound very simple even though it took 13 months of negotiation, but at that point we decided, a small group of us decided to spin out a separate company and that’s how ParcPlace Systems came about. With the deal, I mean we had a business plan where the CIA would be less than ten percent of our business and we were always able to maintain that. Nobody but me ever had clearances so it met everybody’s needs because the government wanted commercial off the shelf products, so it was perfect. There wasn’t a problem there and we started the OOPSLA conference [Object Oriented Programming, Systems, Languages, and Applications] in 1986 in the ACM [to grow the community discussing object oriented systems]. I was President of ACM at the time.

Mashey: Yeah I was going to ask you about that. You should talk to them about the ACM involvement at some point there.

Goldberg: Yeah. And so the CIA would come to the trade show and show everything they were doing. It was not black, which people felt pretty good about. This is how I learned never think about what you’re going to do in terms of, “God what will happen if we fail?” Failure is easy. The problem is if you succeed what have you brought on yourself?

Mashey: You may have to support it, yes.

Goldberg: Well you have a big deal. We built a large company around the world, a big community of companies around the world and that’s a lot to take on.

Mashey: Yeah. Some of the folks who did some of the original Unix work had some of the same issues because they loved to see their stuff get used but they didn't really want to be in the game of supporting it, yes.

Goldberg: And see I think that's not possible. I think that products are successful when there's passion around the product, when you've done something that you're interested in and you care about and to just say somebody else should do it—that does not work. It may be good. It may be bad. But it might not be what you want but I don't think that is the message you want to give to the commercial sector. They want to know that the people who were the inventors cared about it. And I think that we had plenty of competition, other Smalltalk systems, other companies, but I think that the fact that we carried that aura of we were the core team. We weren't all. There were plenty who still stayed at PARC who really were more researchers but we were a core team of people who they could trust had the ideas. And when you're doing libraries, this was true in the Unix world too, when you're doing libraries that people depend on they have to believe in you to have taste.

Mashey: Oh, yeah sure, yeah, yeah.

Goldberg: And, of course, there were very few people from the research team who joined up. It's just that the Dallas Development Center guys moved to Palo Alto. Alan Schiffman came from Schlumberger. We had 25 people in the company before we had finished the negotiation for exit because the President of Xerox said, "Yes, you can do this. Get started."

Mashey: So now had there been previous spinoffs or there were certainly things that in effect became spinoffs but not really? I'm thinking Postscript.

Goldberg: There was no previous spinoff and there was no subsequent spinoff but let me say that more clearly. There were no previous or subsequent venture backed spinoffs. Previously, there had been some joint ventures, 50/50. Spectra Physics was one. There were several more out of the physics group. What you're thinking about was the formation of Adobe but that wasn't a spinoff.

Mashey: Oh, I know.

Goldberg: That was people who just departed.

Mashey: Yeah, yeah, yeah.

Goldberg: And they based their technology on technology they licensed from Evans & Sutherland so Postscript comes from there and not from Xerox is my understanding. There was a company formed to make the mouse. There was a company formed to make the Altos. In fact, I was going through materials recently and found a nice letter from the man who headed up that company when he years later saw my picture in Forbes. So there were lots of these companies but they weren't Xerox companies. When we were trying to spinout, Xerox had an "innovation board" that you went to and you presented to. They were responsible for saying, “Oh you have an idea. We would know if there’s anyone in the company interested in that idea. If not, we’ll make a decision.” So that’s how we got such a quick decision. And we negotiated for a long time period, which translated to payroll cost to Xerox, which translated into equity that the venture group was given. After us, the Xerox venture group, which was solely funded from Xerox, were told “no more.” Whatever they made they can reinvest but they weren’t going to get any more funding. They had funded things like Shugart that Xerox then bought and that was kind of a trade between Apple and Shugart I think. I’m not 100 percent sure. After that there were a bunch of companies that you would say, “Well those are spinouts.” So it’s my understanding that, for every one of them, Xerox retained an 80 percent ownership in [and was the only funder]. There was a whiteboarding company [LiveBoards]. There was a virtual rooms company [PlaceWare] that eventually I think Microsoft bought and there were some ones that didn’t make it. I guess the whiteboard company didn’t make it either although it was a great idea, and there are other companies that do it. But those were not venture backed.

Mashey: Sure, yeah.

Goldberg: The employees didn’t have enough equity to keep them interested. No, it wouldn’t work. So we actually left Xerox in, I know the exact date, it would be March 18, 1988. It was the day we signed all the papers with everybody and it was also complicated because Fuji Xerox in Japan had rights to any commercialization.

Mashey: Oh, yeah.

Goldberg: So they became our partner in Japan. The reason I know the dates is I know the date that Apple sued Hewlett-Packard and Microsoft. That was March 17th, St. Patrick’s Day. I had a board member, an old friend of yours Ben Wegbreit, who said to me, “Adele when we finally exit I think it’s important that we have an article in the Washington Post.” And I said, “But, Ben, those are always little pictures of birds and animals.” He said, “No, you should have your picture in there” and he got his wish because there was an article about the lawsuit and how Xerox had just spun out the technology.

Mashey: Oh, yes, yes, yes.

Goldberg: It was just a very coincidental situation but he got his wish for his little article. It was very funny. You never know. You please people in ways you don’t even expect.

Mashey: You never know, yeah. Well so that was certainly a shift to running a company. Talk about that experience because that’s different than hanging out having a good time in a research lab.

Goldberg: Yeah and with some hindsight it may have not been the best thing to have done. No, I’m just teasing. So I had a number of options in the early ‘80s. I had gotten myself involved in ACM and I had been on the special interest group governing board and that came about because when I was at Stanford one of my Stanford colleagues had encouraged our getting involved with ACM as an "obligation" so I started out more with the special interest group on computer uses in education. I went to a meeting [of the Special Interest Governing Board (SIG Board)]. The chairman of the SIG Board was local here in Palo Alto and he asked me for lunch and asked me if I'd come on the Board. And I said, "I bet you don't have any women on your Board." And he said, "That would be correct." And I said, "And you're asking me because you want a woman on your Board." And he said, "That would be correct." And I said, "I'm going to say yes because you were honest" and it was great fun because I met a lot of people in different fields and got to learn more about all the computer science specialties that I would not otherwise [often keep track of]. Unless you go out you don't always get to touch with all that.

Mashey: That's right, yeah.

Goldberg: And Addison-Wesley, a lot of them had written books with Addison-Wesley and they used to take them all out to dinner and wine and dine, so I'd be invited. I'd say, "I'm not one of your authors. I don't have to come." But then when we went and did the books I knew the publishers and they had an in. They were smart. In '82, I became the ACM National Secretary. Oh, no. So in '79 I became the editor-in-chief of ACM *Computing Surveys* and off of the SIG Board because I went onto the Publications Board so I learned a lot about computer science academic politics, reconfirming how much more fun it was at PARC. There's just so much more to being a professor than teaching a class and doing your research. And then I did that until '82, when I became the National Secretary (when the President was David Brandin who had been the chairman of the SIG Board). And then I became President at a time when ACM was having financial issues where what they were budgeting to spend was in excess of the revenue coming in and for some reason they didn't understand. That's easy. You just don't spend it. But they needed to re-look at their finances. And it's funny because I was the second female president. Before me was Jean Sammet. A decade before she coped with exactly the same problems. She came in really to worry about cleaning up and I don't know why, why other people didn't do it or it was coincidental right? But we had brought a new executive director in and we did some organizational changes. So I got some [executive management experience.] I learned what happens when you have a lot of people working for you who you can't fire because they're volunteers.

Mashey: Yeah, right, right, yeah.

Goldberg: What you really have to do is figure out how to engage them in good projects, how to get rid of projects that really were not going anywhere, and how to leverage the positive. So I was getting some very good organizational learning. I had had some other offers to start a research center for one of the computer companies and I felt like the book was just coming out. I was President. That was enough. I had this new lab [at PARC] and we were going to build the Portland lab. That was a handful. I had kids at home. That was a lot. But I think that ending up in business was not something I gave deep thought to. It was like I knew we had something important that met the goal of empowering people but we didn't understand enough about what this [software] did to empower people, and we really needed to learn and the only way was to just get out there commercially.

It was kind of nice because I was ACM President in those years when a bunch of the guys who were part of the [Smalltalk-80] implementation [wanted to start a conference]. Glenn Krasner, who became vice president of engineering at ParcPlace, was an engineering manager at PARC. And people from Tektronix and people from other companies all had lunch one day working on the implementation and said, "We should start a conference." And so Glenn came to me and said, "We want to start this conference." And I said, "Well we have a little problem. One problem is the budget cycle for conferences is gone. This should be a [ACM] SIGPLAN conference and the budget cycle is over. Another problem is that ACM doesn't really have money to spare so we can't lose money so I'd have to watch over you very tightly that you didn't lose any money." He said, "Well that would be okay. How do we do that?" I said, "Well instead of your grand scheme of having 1,200 people at this first conference how about we plan for 600 and then if it looks like it's going to be more we'll increment it. We'll just run double budgets." So they agreed to do that. And I said, "Well, the President has a little slush fund so I can do this as an ACM, not as a SIG conference, as long as you agree that the next year it becomes a SIG conference if SIGPLAN and/or SIGSOFT want to take it" which of course they did because there were 1,200 attendees! It's now transitioning into a new kind of conference but it's still running [almost 25 years] and still very popular and it's been a great forum because from the outset it was a great mixture of commercial and academic and a nice way to share. So the OO [object-oriented] business experience, because we were educating the commercial sector, had a really important relationship to the academic because you had to change what was being taught. You had to provide trainers and tutors into the companies. We were targeting COBOL programmers because they were the right ones. They understood the business.

Mashey: Sure, yeah, yeah, yeah, right.

Goldberg: And we had to convince the companies, so we trained them and it was actually pretty straightforward in this country but in Asia it was really an exercise to do that. I remember giving lots of talks on that topic [transitioning COBOL programmers] alone. So the conference was good. I actually ended up chairing the 1987 one. I was Past President and I wasn't doing much as Past President. Professional societies are a really critical part of the computer industry whether it's the IEEE Computer Society or ACM, whether its standards work or these conferences, these forums for communication. It's all the journals. I was lucky. I got to taste all of that. So it taught me some things but it didn't teach me how to deal with venture capitalists. Raising the money, that was hard. That was a whole new thing. I was fortunate to have a good mentor in a gentleman in the Xerox venture group who basically believed in what we were doing [Richard Hayes]. Unfortunately a year after we started the business, he died suddenly.

Mashey: Oh, no.

Goldberg: So I think for me that was a bit of a loss because he was one of the few people I felt pretty able to talk through all the politics with because he was kind of fascinated with the whole thing anyways. In the end, people ask me, "Well was it because you were a woman? Was it harder?" And I go, "No, no I don't think so. I think it's just as hard for the guys but I think they laugh it off faster." So you just have to realize that the ridiculous questions you sometimes get asked, not always but sometimes, get asked [both men and women are asked], as in "What kind of car do you drive?" And as you well know, thank goodness, I was driving an eight cylinder black T-Top Mustang at the time, which was the right answer because it was fast, it was powerful and somehow or other that was the psychology they were looking for. I mean it was kind of funny. Or they'd ask, "What were the conversations you had at the dinner table?" And I said, "Thanks to my father I know everything there is to know about the Teamsters Union from management's perspective" but it's not the same as what they were expecting. They were trying to figure out what you knew about running a business. It's hard work to have a company. It's day in, day out, and it's a lot and I think I could have had a little easier time of it in the '80s and '90s if I hadn't done that. But at the same time, I learned an awful lot.

Mashey: So talk some more then about how ParcPlace progressed and what sort of customers you found and what challenges you had then because that is a big shift from a company research lab to running the company.

Goldberg: Yeah it was, but as I listen to what I've told you about the story unfolding because I've not done this before, I realize that a lot of what I was doing towards the end in the '80s was trying to tie our work more and more to Xerox and its business needs, and first and foremost that's what you have to do. You have a product. You're trying to sell it and you need to understand your customer's business and be able to understand their problems and be able to explain how your product solves their problem.

Mashey: It was going in that direction clearly, yeah right.

Goldberg: Yeah. Jim Davis, who was the initial sales guy at Sun, is a friend of mine and he had said something that really sort of changed my whole marketing aspect because what I thought was powerful, what I think is powerful, about object technology is the extensibility, the flexibility of change and the ability to predict what's affected by the change—the entire reuse model, where you can essentially firewall and understand what has to be retested gives you some assurance and some predictability in making changes to large systems and building systems that can change, whether they're for educational reasons or big financial insurance companies, investment bankers (banks on Wall Street, a lot of our customers), manufacturing operations, car companies (Chrysler, BMW, Mercedes), regardless, they all have change as an issue. But what Jim said to me was, "That won't work and it's not working." And I said, "Why not?" He said, "Their problem is they can't get the systems done in the first place so they're not worrying about maintenance. They can't get to maintenance." So we had to switch because it's a two-phased problem: [development then maintenance]. So we marketed with a focus on rapid development, based on reusing tested objects from our libraries. Actually I watched in horror as our marketing people started counting the number of class definitions in our library and selling to quantity, which turned out to matter.

Mashey: Yeah, sure, the thud factor yeah.

Goldberg: But if you just want quantity you can build your libraries with quantity. That wasn't the point but understandability [and thereby reusability].

Mashey: And sales.

Goldberg: Yeah, yeah, but remember these are the years where people were just learning to do graphical interfaces. So as you watch and see what people do we had two big issues, the back end and the front end. So you have this system. Now you need to make it easy to construct the front end. We had a good metaphor, a good set of objects for doing that and so we turned to graphical programming, to providing graphical construction kits to create your graphics. And that's when the product, which was called ObjectWorks, took off. (I remember hating when I first heard that name, but realize now that Doug Pollack, who was the marketing/sales guy at the time, was a genius. I mean it's a really good name—its Objects and it Works.] The new product was called VisualWorks because it allowed you to create the visual front end, and then the business took off because we solved the front end, rapid creation problem. The back end problem was database hookup. In the research lab, we worried about databases but we worried more about objects, finding objects, an object oriented database. In fact, Bob Flegal worked a lot on that [including building a weather monitor to notify him when to go windsailing on the Bay!], but we didn't try to commercialize any database system. There were other companies that were doing commercial object-oriented databases.

Mashey: Yeah right.

Goldberg: But if you're going to sell into legacy worlds, the thing they put the most design behind, the aspect of their systems that is the most critical, is the design of the SQL tables. I know how critical this part is from the last decade of work that I've done, mostly dealing with exactly this design problem. We were fortunate that we had vendors who sold kits for mapping objects to SQL databases, but we hadn't done that work ourselves. There was some ugly fighting going on too. You'd think you were in kindergarten, not even in college. The fights between the structured programming methodologists and the object-oriented methodologists, were pretty bad. I finally realized that the structured guys thought we'd be putting them out of business and that was one of the reasons for the unbelievable emotions, terrible emotions. That was one. The other fight was especially in the government sector. I had somebody from one of the government contractors tell me I was awful. I was ruining their business. And I said, "Let me see if I can characterize your business for you." We were down in Florida. We were behind the scenes at Disney World where there are conference sites and we were in the hallway and this guy was on the attack. I said, "Here's what I think your business is. I think you pitch to the government a four-year project with 400 people that you know that you'll be able to fund for four years, and you under-bid it. You have to get more money but they're four years in so they give you the fifth and the sixth. That's your business." He said, "That would be correct." I said, "Then I'm going to ruin your business." And he came into the meeting arguing that you can't use an object-oriented approach because it means you're reusing and if you're reusing you're not meeting their security requirements. He was referring to the need in the intelligence community to compartmentalize and he was claiming that the names of the objects and their messages were part of the secret for each development project.

Mashey: Oh, boy.

Goldberg: And fortunately the man in charge, one of the group chiefs said, "Well we're just not going to do that anymore or we'll alias it, but we're going to reuse. We can't afford this any longer."

Mashey: Yeah, yeah.

Goldberg: So I mean those are the kinds of things you learn about when you have real customers. The Analyst Workstation was a very important application, with advanced spreadsheets and information editors [targeting intelligent information retrieval and document preparation]. One of the Wall Street guys built a whole framework in which you could design financial instruments. I know from these two cases alone that we were making a difference. I used to say "They're spending millions between software and training and consultants helping them build. I sure hope this is worth it to them." And every one of them told me that after ten minutes' deployment, they got a ten times return on the investment. Talk about learning. Millions means nothing when what you're playing with is really a lot more, what's at stake is a lot more and some of it is qualitative. There was a conference on the West Coast that Steve Jobs gave a talk at and he was pitching object technology reuse. It was funny to watch him pitch because I had been behind the scenes, excruciatingly, painfully teaching the magazine guys, like writers for *BusinessWeek*, what the good examples were and what objects were, because I couldn't afford, even though the article wasn't about us, I could not afford to have it be wrong.

Mashey: Yeah, right.

Goldberg: So Steve got up and he says, "Well, you wouldn't take an existing system, like you wouldn't take your existing payroll system and rewrite it all. That makes no sense. It's going forward with new applications." Then he gave his usual good, fun talk. The next speaker was one of our customers from Chrysler and he got up and he said, "We are rewriting our payroll system" and the reason is that we have to get rid of the inflexibility of our current systems. Basically you can watch Chrysler as probably one of many examples [where management is] having a poor relationship with the unions because the unions would make what, in some perspectives, were perfectly reasonable requests in terms of compensation changes. And Chrysler [management] would want to say yes, but they knew they couldn't do it because they couldn't implement it. They couldn't change their systems fast enough. So what they wanted was to be able to say yes to the unions by having a flexible, extensible system. It was one of the best examples I had ever heard for why objects. That's what I meant by finding out what your stuff is good for, when your customers tell you this type of stuff. So we were all learning. So in a sense the business [like the research lab before it] was still a learning environment. I think that's what sustained me through all of that effort. After VisualWorks, then we took off enough, four quarters of profitability, and we went public. I had hired somebody else to run the company. He and I didn't get along very well so I left. Oh, then we merged with one of our competitors, with Digitalk and I could see the company was going in a direction that wasn't going to be fun for me so it was time to leave.

Mashey: So how long was ParcPlace for you?

Goldberg: I think I stayed on the Board until '95 and it was only a short while later that the company was sold. For me, I learned a lot about Board structures and how complicated it is in a venture-backed Board when they [the investors] wear multiple hats. There are times when they won't take actions. There were a lot of issues there. I've been fortunate. I brought that to other companies where I've been on the Boards now and kind of learned a bit about the role of a Board member. I don't like it when, other than the CEO, there are inside people on the Board. I think insiders wear an extra hat [which makes it hard to focus on the Board-specific job].

Mashey: It's hard, yeah.

Goldberg: The venture guy has his partners and their annualized valuations to consider as well as the company longer term perspective. Those are two hats. I think that's complicated.

Mashey: It's hard, yeah.

Goldberg: Because you [as an independent Board member] always have to figure out what they are worrying about? What's really going on?

Mashey: So then what was next after that?

Goldberg: Well then we started a contract research activity through Mitsubishi, which was one of our customers, well Mitsubishi in Japan, but managed through Mitsubishi Research in the Boston area. They were interested in the virtual community problem [how to support project teams who worked in remote locations]. This particular problem was an extension of one of the big outcomes of doing ParcPlace, which I think was a bit of an error on our part not to understand up front. When you are in research or you're thinking about a research task, the nature of the programming situation is clearly different than when you have a large and diverse team. Two people can do fine, regardless of their location. When you get to three and more, then you have co-management and coordination issues.

Mashey: Sure, yeah.

Goldberg: And that's true in software. There were good solutions in the Smalltalk world done by another company. Envy was an example of how to coordinate software development by members of large teams. We were doing our own in-house work funded by some of our customers because the Envy approach was one particular style and we had some other ideas. But if you extrapolate from that particular product and look at the coordination problem more broadly, you easily find that it doesn't matter how clever your language is. It doesn't matter what your libraries are like. It doesn't matter what your visual construction tools are like. In the end, there's going to be a team. They're going to need to collaborate. They need to have a shared vision. I had written a book, done some studies and written a book with Kenny Rubin on specifically this problem [*Succeeding with Objects: Decision Frameworks for Project Management*]. We got the book right but we got the steps in the wrong order in terms of what you do but that's part of the learning. And so it was that next we wanted to understand, "Well it's hard enough when they're all in the same building. What happens when they're in different cities and it is very costly to travel frequently?" And we had learned a bit in the Xerox experience that you really can tie people together in multiple cities in a way that felt fairly close and have a sense of team [the research lab had the teams in Palo Alto and Portland working on common projects]. There were some tricks to it but you could do it. Mitsubishi was more about manufacturing and engineering development and they were interested in the problem, so they offered to fund R&D. We received $4 million from them to do a multi-year research project and so we actually designed and built one of the very early online project management systems. It was a little bit too research-y, a little too clever for what was ultimately commercialized. But at the time we finished the work with Mitsubishi, I said, "I can't do this [build a commercial company] again because there isn't a waiting customer. It's not a pull market. It's a push like it was for object technology. It's too much marketing. We're too early again." [This work was pre-Internet.] And I just didn't want to make that into a company.

I had at the same time concluded that the primary interest in these kinds of systems (where it wasn't just lists of things that were being checked off but serious collaboration) were going to work best in worlds where there are compliance obligations. So the reason is that you're asking people today to spend time almost anally logging all the details of their work [planning and acting], interacting online. I mean obviously you can use the tricks we have now, everyone does, with discussions. There are online discussion forums but you discuss via email and the messages are archived and that facilitates some of the information capture because our generation is used to email. The current generation "Twitters" so it would work even for them. It's amazing. They can say so little in such a short amount of text— they can say so much in such a little bit of space. I just didn't want to push structuring projects and capturing detailed actions at a time when the market was not demanding such systems as yet. The compliance systems were for waste management and pharmaceuticals, [and here the current team members know that they can only be successful if they can prove what they did and did not do]. And I had met somebody in the pharmaceutical [industry] who started teaching me about it. So I've actually for the last decade been working with that same person and ended up working with a private equity fund in drug development, building some of the initial [tracking and coordination] systems. It's like a new career learning all about how drugs are developed and I'm enjoying that.

Mashey: Cool. Anything we haven't covered that should have been because you've done a lot of different stuff?

Goldberg: Well, I suppose if you were Alan Kay sitting there you would be shaking your finger and saying, "What happened to your interest in education?" So I've tried a couple things. I got involved with a company that was doing mostly CDs, very high quality, and it didn't make it. It was like too late for CDs and too early for the online, something like that, but the executives there got funded for and started a new company to do online delivery of educational materials. I understood that what they were trying to do was build a sense of community amongst teachers and I was interested in that. So I was their initial CTO development manager and ended up being there for four years building an authoring and delivery system, becoming a Python programmer.

Mashey: Oh, yeah okay that was the Python project.

Goldberg: That was with Python [using Zope] and it was lovely. I was really very happy with the system. You have to get your head around it but once you do you can build a lot fast. I swear we had very little time, counted in terms of months, less than a year to build an authoring environment using XML which was pre-processed to generate the HTML. I used Smalltalk for the preprocessor. Then we built a learning deployment system and spent a lot of time mostly in Texas in the schools down there. But in the end we didn't really do teacher-community. The politics, I kept being told the school politics doesn't allow it. There were a lot of those kinds of reasons although I'm not convinced. The New York Times Company currently has a new project in this area. They seem to be going after what I had hoped for. Now a number of years have passed but we ended up doing more electronic books online. I think they're effective as classroom aids to the teachers but they're really not for the kids and they're not for the teachers to help one another, So after four years it was stable enough and I left. They needed to rebuild everything. [Two and then three of us for almost 5 years did all the prototype and deployment in Texas, NY, Illinois, California, with several secondary and middle school math courses. We had only 5 months to build the initial systems to be on time for teacher training the first year of deployment, and, given all the new functionality we had to deploy, we were never able to go back and throw out that first system. Eventually the company hired 8 or 9 people and gave them more than 3 years to get a comparable result engineered.] So I moved on from that to focus on the drug development ideas.

And then I started thinking about what else would you do and how would you do it? I looked at the One Laptop Per Child activities and I talked to Alan but I haven't found the right thing in education to go back to. I've talked with a number of people about one area, where the problem interests me. (I think I've mentioned to you before.) I would like to get my head around a curriculum that says you are a kindergartner; what should you do with computing that would lead you by high school to be able to use computers effectively as a thinking partner? [How might our educational experiences differ if we designed the educational system today, with no requirement to use any legacy curriculum and with the belief that literacy should involve computing-based technologies and the expectation that our knowledge and skills will continually change, rather than define literacy as being pencil/paper/book-based with the expectation that we are learning up to a specific age when we emerge as learners and become doers.] I've spent some time on thinking about this problem, but I don't think I have a good solution yet. I'd like to because I have grandkids and I'd like to get them started.

Mashey: Yes.

Goldberg: But I haven't figured that out yet. So I mostly sit on Skype with my 6-year-old granddaughter and I invent word problems for her to solve rather than just, you know, doing arithmetic—3 plus 2. It's more fun to make up stores, like pretending she has candy and has to divide the candy amongst her friends. She likes that better as well, and can make up math stories as well.

Mashey: Okay, anything else?

Goldberg: No I think that's it.

Mashey: Okay, so let's start with telling us a little bit about doing systems research in the '70s at PARC. What was that like?

Goldberg: So to talk about '70s at PARC I have to back up a bit and just note what I had done before that. So I had been working in more traditional systems, building on an IBM System/360 at the University of Michigan or an IBM 704 at the University of Chicago. We were actually building our own machine as well, but otherwise worked in very traditional timeshared, multi-user type environments. And then I went to Stanford and we were building applications that assisted children in learning math, mostly math. I was involved in teaching elementary level and college level symbolic logic using a PDP-10 timeshared system. I had to work at night, all night long, because the kids were on the machine during the day. Obviously the dream at Xerox PARC and the reason I went there was to give more ownership of systems to the users and we were interested both in kids as well as adults. So the idea that all the power of the machine was going to go to one person, and you could think about it that way rather than spending all your time being practical about time and space, changed everything, just your whole approach to what you were doing. Secondly, I was used to funded projects, mostly NSF-funded projects. When you do funded projects, you've done a lot of it before you receive the grant monies. Then you get the grant and you have to get it done and you have to get more done so you can get your next grant, etc. And so you're always watching what are you going to publish, what grant will you go after, what is being funded? And Xerox was such a gift. We estimated you had about five years of real exploration, real prototyping, really doing something, learning from it, throwing it away and starting over again. And so that was really very special, very different from the mostly university experience that I had before going to PARC.

Mashey: So you, of course, are a software person, not a hardware person particularly.

Goldberg: Correct.

Mashey: But tell us a little bit about the Alto and when that came and what that meant for you folks.

Goldberg: So by the time I got to Xerox PARC, there was a first Alto but if I recall the display screen was covered with cardboard. It really wasn't a finished product in any sense. We thought of it as an interim Dynabook. So obviously I went to Xerox to work on the Dynabook project whose arrival time we miscalculated. We thought we had about five years of hardware development before everyone would carry a high-powered computer as powerful as a PDP-10 wherever they went. And so, of course, we underestimated the power of what you'd have and we certainly underestimated the timing [but only by 6-10 years]. So we viewed the Dynabook challenge as primarily a software problem and we treated the Alto as an environment in which to explore what that handheld computer would be all about. We had only a few of them and they were designed and built in the Computer Science Lab with Butler Lampson and Chuck Thacker in the lead. Obviously, Alan Kay was very involved in specifying requirements and I think in providing some of the funding. My responsibility was really at the user level. What are people going to do with the Dynabook and, in that early case, what were they going to do with the Alto?

Mashey: Is that where the Smalltalk work came from?

Goldberg: So, right. Since we assumed the hardware problem would be solved, although we didn't quite assume without some serious nudging that the flat panel display would be solved and we were involved with Anne Chiang for example in one of the other laboratories who was interested in exploring technologies for flat panel displays. But basically we saw it as a software problem and we saw it as a software problem for how can you enable the average person to use a computer to communicate, to communicate with themselves as well as others. Communicating included computing. If you're going to do computing and you're going to own it and make it yourself, we believed it meant that, at some level, you were going to program. So it came down to coming up with a software idea and a software approach that allowed you to program at all levels of the machine and that what the professional programmer or system programmers did could be exposed at some point to those who were more interested in what they were doing than programming per se. So you wanted some uniformity at all levels. And so when we say Smalltalk we mean the language of the Dynabook and it had many, many iterations through the years. The very first Smalltalk in 1971; when I got to Xerox, Smalltalk-72 was already started because I got there in 1973 and every two years we did a new implementation and then the alternative two years we did a new language, or rather simulated a new language on that implementation. So there were many, many versions of Smalltalk until 1980, at which point we felt that we needed to share what we were doing with the world a little larger and get some help with what was turning out to be a very hard problem.

Mashey: So this is all old stuff for you but as I recall some interesting new things really got going in that environment in terms of user interface.

Goldberg: You know it's really interesting because so much of what was done we almost took for granted that there was no other way. I think at some point you get so in the middle of what you're doing, what your research is, that you forget it isn't what everybody else is doing. I had been on graphical displays on an Inmac display on the PDP-10 and an IBM 1500. We had done graphical interactive systems but not with the flexibility of the bitmap display. So amongst the things that I think were very innovative was not just *that* we did it, but *how* we did overlapping windows. The problem being solved is, no matter how big the display is, it's too small. How can we make it [virtually] bigger? The second problem is no matter how much you try to make the system uniform, consistent, simple to use, if you're programming you're going to have to find out how to do things. So the entire user interface metaphor was about search and that's where the browsers came from—searching for code, searching for relationships amongst the parts of the code. Smalltalk is an object-oriented system so code is organized in terms of objects that own their own data and you have to ask the objects what to do by sending them a message. So a lot of the query interface was very much about who's doing what? Who knows how to do what? Who can I send this message to, to get something to happen?

Another thing we did I think was fascinating was to be able to interrupt a running system and say, "Who's doing that?" Now I say "who" when I really mean "what"! I mean what object in the system is doing that. But it allows you, when you know nothing about the system and how it's really put together, it allows you to see things that you're interested in and be able to find out how it was put together. And the third thing that I think was innovative was the approach we took to multimedia. If you're going to communicate, you're going to communicate with text. You're going to communicate with static pictures. There's going to be animation. There's going to be executable code and they're not going to be independent. They're going to be very well integrated. One of the loveliest inventions, which I think comes first from Larry Tesler and followed on by Diana Merry, was to be able to have a document showing all aspects of those different kinds of media and by touching one kind of media, an appropriate editor would come up and you could change it in place rather than taking the approach of opening an application, copying things in, making the change. So it was a smoother way of interacting with the system.

Mashey: So a lot of folks have used the term WIMP. Did you folks ever actually use that for Window Icon Mouse Pointers?

Goldberg: We didn't talk that way, no.

Mashey: You just did it. You didn't talk that way, yeah.

Goldberg: Yeah. I mean we were obviously doing that and we were doing it with music. We were doing it with animation. We were very interested in how to sync them and how young children could manipulate the information behind music and behind animation, but we didn't coin or use that terminology.

Mashey: Okay, but how about in terms of influence? Perhaps talk a little bit about looking back now, the influence of the work on the Alto on the current evolution of personal computers.

Goldberg: So there's a hardware influence and there's a software influence and the software obviously is programming language as well as applications. I'll start with the hardware. If you look at the chronology of hardware that was built from the very first Alto, Xerox was very public about what it did, we had visitors all the time. Sometimes we didn't even realize a person was a visitor because they were escorted by some other Xerox employee and I don't always know the rules on how people entered the building. But clearly there was an influence into Carnegie Mellon and into the PERQ computer that came out later. Our own work influenced Tektronix when the 4010 [graphics terminal] came out. They even put a Smalltalk system in their oscilloscope which is an interesting idea in terms of embedded systems. And, of course, there's the infamous argument about who was stealing what from whom in terms of Microsoft and Apple. Apple from a hardware point of view, clearly along that chronology you see the Lisa and you see the Mac. I mean it was clear that the Alto, which then progressed into what we used to call the D machines, so the Dolphin, the Dorado and the Dandelion. The Dandelion was the commercial machine out of the Xerox Office Systems Group. They were obviously flowing from the ancestor of the Alto itself with the same people involved in the design. So I think that's the hardware one. It was the granddaddy of powerful, individual workstations that I used to say to the students, (the kids I taught, mostly junior high school who'd come into PARC for our resource center and we eventually took Altos to Jordan Junior High School to see what the kids could do with the combination), I'd say to them, "The only person [thing] you hurt by trying to break the machine is yourself [well, your work], so go ahead and explore."

In terms of programming languages, Smalltalk is derived from Simula. There are a number of other languages that influenced taking a message sending approach to objects. And then obviously Smalltalk as a programming environment in terms of how you program, not just the syntax or semantics of the language, influenced the interfaces to Lisp and to Mesa. I would believe that that would be considered accurate. And then there was Objective C and C++ and Python and a number of other languages that tried to capture the idea of modularity., We all learned from Simula. What we learned from Simula primarily is that it's a simulation language. The purpose of that language was to enable people to simulate the world as they understood it so they'd understand it better or verify their understanding. And that's what we were doing in the Smalltalk world, trying to make sure that people could build applications [or models] that in some way represented their understanding of how some aspect of the world worked. That was a big part of the commercialization of Smalltalk, which was for businesses to simulate their business processes and be able to step in and assist in those processes.

I've been using some words, including "Smalltalk" and "objects" and "message sending" that perhaps aren't broadly known. Most computer programmers today now know what they are but certainly in the '70s and in the '80s when we commercialized the language, a big part of what we had to do was educate. So Smalltalk is what we call a programming system. It's a language and all languages evolve over the years, but they mostly evolve in terms of the libraries [of objects or procedures] that you can use. But it's basically a language that is made up of objects, each of which you can think of as a simulation of a computer. The computer has data and it has operations on that data, so every object is a way of representing the data to carry out the operations. So if I were simulating a business process, for example, in a financial institution I might represent the various kinds of financial instruments I have and the flow of buying and selling those instruments. I would include customer and customer accounts and how we would report on them. All of the key players or actors in that world would be represented by an object.

Mashey: Okay so what are you working on now? What are you excited about?

Goldberg: So I have sort of a general area that I work on which I think of as creating virtual communities, building communities online. Actually that work started in 1984. Once we had finished a lot of the Smalltalk work and got it out the door that was personal computing and we started realizing that the problem we had actually posed for ourselves was interpersonal. How do people communicate with one another? How do they share their interests? How do they help coordinate with one another when you're working remotely? In the end of the 1990s we formed a company that got funded to start researching that particular problem and it was focused on primarily engineers, but it's evolved now so that the two large projects I'm currently working on have to do with helping international researchers find one another and collaborate over distance, share their resources, hold their conversations. The other area which is I think the one that is the most exciting is in drug development. I work with a group of people who are very interested in how you can run a virtual company, a small virtual company that takes early stage assets—molecules, and takes them through the pre-clinical stage to proof of concept. In trying to do that we only have 20 people but you have 20 or 25 drugs which means technology must enable us to scale.

Mashey: So then that sort of leads to the next question. What do you think the next big challenge is for the computing industry and computing in general?

Goldberg: Right, so besides the interest in the virtual communities, one particular virtual community I've worked on is helping teachers help one another wherever they are so I still have an interest in educational uses. So the two big areas that I really want to see technology focus on is healthcare and the personalization or customization of healthcare, and doing a much better job in the education of our children, actually lifelong education not just our children because once you're learned and learned how to learn on your own and find other people to learn with that becomes a lifetime activity.

Mashey: So how do you think technology jobs will change in the future? It's a lot more ever present than it used to be.

Goldberg: That's a hard question because, of course, it ties to what do technologists do all day. I had at some point scoped out a book for children that was going to be called *What Do Computer People Do All Day: a Scary Story*. To some extent for most people it's still scary. They don't really understand what you're trying to do. We invent languages for specific areas. We invent languages that let us simulate business "language" so that we can allow people who are experts in businesses do their own programming. One of our goals in the '70s, which I think has happened, not as much as we'd like to see but has happened more and more through the '80s and '90s and now into the 21st Century, is that most people feel empowered to change the computer, but the way they do it is still cumbersome. They still have to learn what computer scientists have to learn. If you're going to teach a child in school, you're still going to give them a test and see if they understand about a computer, we still think of it as teaching them a programming language. And then we argue over which programming language and that's not a conceptual way to teach. So I think one of the things that has to happen, whether it will or not we'll see, is that we have to understand more and more what it is a kindergartner learns about a computer, what a first grader learns, what a second grader learns, the same way we think about teaching them to read and write, teaching them mathematics by starting out with simple concepts and building up and not over expecting what they know, so that it becomes a natural tool. When we look around the consumer world, we see everybody with their phones. We see everyone playing with media. I have grandchildren ages six and down and they know how to call me up on Skype and do their homework on Skype and it's all natural communication channels to them. But what they don't know how to do is to take that computer and mold it to their own ideas the way they do with crayons and pencils on a piece of paper. So we still have that as a challenge.

Mashey: So if kids get a good education and they decide they want to go into a technology career, what sort of advice do you have for them about what they might or ought to think of doing and how they ought to prepare for it?

Goldberg: Well I know what I tell them today. Even those who are more mathematicians and who want to do more theory. There are two things they absolutely have to focus on if they want to go into computer technology as an academic pursuit, whether they then stay in academia or go into the commercial sector. One is mathematics, mathematics, and mathematics, I mean we've lost that. We all started doing math. We may have done it through physics or a chemistry sequence, but when I went through school, undergraduate and graduate school, mathematics was at the core. That is not the case if you look at most computer curriculums [today]. And secondly, you have to have something else you know about. It's an application [the use of computers]. The one I would choose if I were back in school is I would learn more about biology because, again, I think one of the biggest problems facing us in this world is healthcare and understanding the science behind our health problems or health opportunities is critical. So being able to apply the computational sciences to healthcare is critical.

Mashey: Good.

END OF INTERVIEW