May 3, 2002

Oral History Interview with Gary Durbin

Philip L Frana, *James Madison University*

An Interview with

GARY DURBIN

OH 368

Conducted by Philip Frana

on

3 May 2002

ADAPSO Reunion Meeting
Washington, D.C.

Charles Babbage Institute
Center for the History of Information Technology
University of Minnesota, Minneapolis

Gary Durbin Interview

3 May 2002

Abstract

Gary Durbin is a software pioneer and entrepreneur with over thirty-five years of experience. He began his career specializing in operating systems and database systems. His first company, started in 1970, developed operating system improvements for IBM machines. That company introduced Secure, an early software security product. Secure was sold to Boole & Babbage in 1978. Durbin then founded Tesseract Corporation, a human resources software company that introduced the Time Relational Database. Tesseract was sold to Ceridian in 1993. Durbin founded Seeker Software in 1996, which was acquired by Web application company Concur Technologies in 1999.

In this oral history Durbin recounts his education at the University of California at Berkeley and early Wells Fargo jobs programming the IBM 650, 1400, and 360 mainframes for online branch banking. He describes his activity in the consulting firm Cybernetic Systems Incorporated, his sole proprietorship of the Institute for Cybernetic Development, and the founding and financing of Tesseract. He also notes his role in the founding of parallel processing software firm Primrose Software and in marketing the Web application system Seeker.

Durbin explains the core characteristics of good programming, software engineering, and management. He describes his work in developing network and relational database management systems, and the hegemony of the hierarchical IBM database system IMS (Information Management System). Durbin also explains the importance of integrity in a business prone to marketing vaporware, the impact of IBM's unbundling decision, and the recruiting and retention of women by software firms. He notes the role of the software industry in jobs creation and in endorsing the Black/Scholes options pricing model.

Durbin also relates the importance of user groups like ADAPSO to the development of the independent software industry, including ADAPSO's financial accounting committee and the special interest group Software Industry Association (SIA). This oral history was co-sponsored by CBI, through a National Science Foundation grant project, "Building a Future for Software History," and the Software History Center in conjunction with the Center's ADAPSO reunion (3 May 2002).

Preface

As part of its preservation activities, the Software History Center (SHC) worked with Dr. David Allison of the Smithsonian Institution's National Museum of American History and Dr. Jeffrey Yost of the Charles Babbage Institute to plan and conduct a number of oral history interviews of early software company founders and other key industry contributors. On May 3, 2002, in conjunction with SHC's ADAPSO Reunion meeting held in Washington, DC, SHC arranged for 15 individual interviews by historians well qualified by their knowledge and interest in computing history.

The following people were interviewed together with the name of their interviewer:

Bruce Coleman, interviewed by William Aspray
Richard Crandall, interviewed by Paul Ceruzzi
Gary Durbin, interviewed by Philip Frana
Martin Goetz, interviewed by Jeffrey R. Yost
Bernard Goldstein, interviewed by David Allison
John Keane, interviewed by Martin Campbell-Kelly
Ernest E. Keet, interviewed by Philip Frana
Frank Lautenberg, interviewed by Paul Ceruzzi
John Maguire, interviewed by William Aspray
Joseph Piscopo, interviewed by Thomas Haigh
Lawrence Schoenberg, interviewed by Martin Campbell-Kelly
Charles Wang, interviewed by David Allison
Robert E. Weissman, interviewed by Paul Ceruzzi
Lawrence Welke, interviewed by Thomas Haigh.
Sam Wyly, interviewed by David Allison

Each interview was tape recorded, transcribed and edited by SHC, the interviewer and the interviewee to ensure clarity and readability without changing style or flow. The original tapes along with the edited transcripts were donated to CBI, which placed the edited transcripts on the CBI website.

ADAPSO History Program
Gary Durbin Interview

Philip Frana: This is an oral history with Gary Durbin conducted on May 3rd, 2002, by Philip Frana. Thank you for agreeing to sit for this oral history Gary. I appreciate it. Would you first tell me about your early life and how that prepared you for the software industry in particular?

BACKGROUND

Gary Durbin: Well, I suppose the most significant thing is that I went to school during the Sputnik age when educators were very interested in tracking people toward scientific careers. I am a native of California. So from a fairly early age, I knew I was going to go to the University of California at Berkeley. It was in fact the only school I applied to. A whole group from my high school had been tracked.

PROGRAMMING

Durbin: I was a year or so into Berkeley when I needed to take a semester or two off to make some money. It was clear from my family situation that I was going to end up putting myself through school so I decided to get some kind of skill. I went to technical school and learned tabulating equipment—that's what it was called in 1961. And I found that just great fun. It was amazing that people would pay me to have so much fun. So I got a job at a bank. I was a natural. One of the instructors recruited me out of the class to go to Wells Fargo Bank, which is where I started, and within 9 months the woman who had been my supervisor worked for me because it was just this natural thing. It was mind candy for me.

I learned programming on the last generation of tube computers, the IBM 650, and spent the early part of the 1960s converting those programs and building new applications for the 1400 series: 1401s, 1410s, 1440s—that class of commercial equipment really exploded. IBM made the 1401 and they expected they'd sell perhaps 200 of them, but actually sold ten thousand or something. It was a phenomenal thing. By the time the 360 series came out I was well known in the company as one of the technical experts so they gave me the new operating system stuff to work on. It was incredible fun and I worked really long hours simply because it was like a drug. You started taking that stuff and discovered you had to come in on the weekend and finish up what you were working on. So I worked on operating systems in the banking industry. It was all high performance check sorting and telecommunications for branch banking and that kind of stuff.

Frana: All batch at first?

Durbin: Oh no, I was doing real time systems.

Frana: By the time you were working on the 360s?

Durbin: Yes. The trick with check sorting systems is that they are very high performance-oriented systems because from the time you read the check, you've only got a certain amount of time before the check is moving and passes the pocket that you have to drop it into. So we'd get down to counting instructions and figuring out which instructions ran faster than other instructions so that we could get the pocket selected before the check missed. If we missed the pocket it would go to the reject pocket. And so this was really high performance, super real time critical stuff. From there I got into telecommunications systems for online branch banking.

Frana: What makes a good programmer?

Durbin: Programming is an art. I really feel it's an art. I've known a number of people who have been fabulous programmers and part of it is being able to see structures. It's not about lines of code and so forth—it's about the structures, seeing how the structures all fit together. Some of it is control blocks and some of it is objects and how objects relate to each other. So if you can get to that level of abstraction then you can begin to see how software ought to work. It's not a matter of coding from the specs, it's seeing how it ought to work, how things ought to fit together. A lot of software is also about trying to map reality. Particularly in databases, we try to model the structures of reality. So if you're building an accounting system you have to understand the component parts of the organizational core, and the concept space that you're working in. If you can get to those levels of abstraction you can talk with the people and understand the system and begin to see what it is they don't see. They see the transactions moving back and forth and you see the structures that the transactions are moving in and affecting. Then they ask, 'What is the nature of those transactions?' 'Are those time relationships?' 'Are those entity structures?' Whatever they are, so you can abstract. It's about being able to abstract reality and then model the software system over that abstraction of reality.

Frana: Has this changed over time? Or has that always been a characteristic of the very good programmers?

STRUCTURED PROGRAMMING

Durbin: Well, in the 1980s there were structured programming models and you know that shifted. That began in the mid-1970s and was going fairly strong for most of the 1980s. In the 1990s I was moving to object-oriented design, which is another word for the same thing. Actually, I went to one of the first sessions in object-oriented design—it was given by the same people who two years earlier had been doing the structured programming stuff. They had just gone through their slides and changed all the names. The process is still about the same. It's trying to understand the abstractions. That's what these systems are all about: structured programming, object-oriented design. They are methodologies to try to provide vehicles for getting to those abstractions and then the tools and mechanisms for understanding those. Like understanding the message protocol betweens objects; as soon as you begin to do that you begin to say, 'Well, wait a minute. This isn't the solution.' The problem is that design is an iterative process. What people tend to think is that you start here and you do these things and then you go there. That isn't the way it works. You go here and in the process of design you discover things and then you circle back, and then at some point you just go dig detail, detail, detail, detail. And then you start moving back into the

abstraction layers. The processes aren't really designed without it. You have to have a sense of how that works.

Frana: So even before it had a name, the people that are doing it effectively are doing it as an iterative process?

Durbin: Yes. People that I know were doing it back in the 1960s and the 1970s. I remember one of the guys that I worked with who was an incredible programmer. I mean he could write several-thousand line programs that would run perfectly the first time out. He was one of the few programmers who wrote programs in ink, back in the days when we had to write them out and then go key punch them. He'd write the programs in ink because he had it so well thought out in his mind that it was only a matter of getting it down on paper. It was not a matter of crafting it. And some structured programming classes came out and he went to some of those and he said, "So what else is new? Of course, this is obvious." For those people who have that sort of structural bent, it was obvious. Of course, we tried to create a software discipline and I think that there has been a lot of really good work in software engineering to turn some of those methodologies into disciplines so that people can get to that place faster.

Frana: So they can be trained so that they don't all have to be like your friend?

Durbin: And have to discover it by themselves, yes. My coworker was programming in the mid-1960s. I started in 1961. I think he probably started in about 1965. It was much more difficult in some of the earlier systems, particularly because the systems were batch, to get to some of these structures since you couldn't isolate them. You had to put everything in flat files. I remember I went to a class. It was the first 1401 class. Somebody was explaining how to understand magnetic tape and they said that magnetic tape is just a bunch of cards with scotch tape in between. Well that was one way to think about it you know, still 80-character records but strung out so that you could read them. It really wasn't until we got to disc systems where we could actually begin to functionally organize things. In the early days you would develop a program and all of the data was on a single tape file because the machine only had two tape drives: one tape drive for input, one for output. It wasn't a matter of saying, 'Well, we'll put this functional data here, and this functional data there.' You just simply couldn't do it. You had the same problem with card systems. When we had card systems there would be a code in the card that indicated it's one of these or one of those. You couldn't isolate things. Later on, with structured programming stuff, it began to structure the code and of course object-oriented is binding the data structures and the code structures together. It's the next progression.

Frana: Did you find yourself coming down on one side or another in the "Go-To" controversy? Did you take a strong stand on that?

Durbin: The Go-To controversy, well sure, because the problem with go-to's was that you couldn't isolate code if you could all of a sudden branch in the middle of it. You couldn't control state. Part of what object-oriented programming is all about is understanding the scope of state so that the state of a particular data section is limited to only that code that is bound to it. Well, of course, go-to would be like branching in the middle of some object and not having any clue what the state was. So part of structuring

code was to not only structure the data but also to structure the components that refer to data and trying to limit scope.

Frana: What were the managerial issues? If you could force your programmers to write in a much more structured way then could you take better control of them?

Durbin: Well, that's interesting. It really went the other way. If you taught people or if people learned what structure was all about, it became obvious. But to think that you were going to get good structure by saying you can't use a go-to state is exactly backwards. There were places where those methods of coding made sense. You know the equivalent of a case statement where when something is true you want a branch or go to the exit of that case. In C and JAVA we have a K statement and if you wanted to code the equivalent of a K statement in COBOL or FORTRAN you needed the functional equivalent of a go-to. And that's a good, good strategy. Just saying that you *can't* do that doesn't make sense. I remember in the mid-1970s I interviewed a programmer for a job. She was a bright woman and so forth, and it sounded initially like she had the skills we were looking for. But we gave people little pieces of code and said what do you think about this? So I gave her something and she said, 'Oh we can't use that statement.' You know, a compute statement or something. And I said, 'What?' But she insisted, 'Oh no, no, we can't use that.' So I dug into it a little more. She had worked for an insurance company and they had all these rules about components of the language you could not use. Somebody had said this isn't a good thing, probably years before, and they were carrying it forward. They also had this process where people wrote code and then somebody else debugged it. Well, that was sort of contrary to the whole concept. How are you going to understand your mistakes if you are not engaged in finding and correcting them? There is a great book on writing called *One Continuous Mistake*,[1] and programming is sort of like that, it's one continuous mistake. What you do is to define the problem as best you can and then through debugging you refine it.

Frana: So it's difficult to create a layered bureaucracy in programming?

Durbin: Yes, I think it's typically counter-productive. It addresses the wrong problem. If people don't understand how they ought to structure code you're not going to get them to structure code well by those kinds of rules.

Frana: You started out using navigational databases in the 1960s and mid-1970s and then you switched over to relational databases technology?

RELATIONAL DATABASES

Durbin: Well I started working with the equivalent of a network database about 1968. We were developing a very large claims processing system for an insurance company.

Frana: You were still working at Wells Fargo at this point?

---

[1] Gail Sher, *One Continuous Mistake: Four Noble Truths for Writers* (New York: Penguin Arkana, 1999).

Durbin: No, I was off in my own consulting company with a couple of other people. Network databases are highly efficient because you find this, you've got a pointer to that, and you can go straight there and there's very little traversing. So we built a network system and we began building some applications that looked more relational. Early on, about 1976, I discovered Codd. Actually what happened was that a friend of mine was over at UC Berkeley and gave me a call and said, 'Hey we're working with this thing over here called System R. You ought to take a look at this.' So I went and visited him and I took a look. Of course, System R was a prototype that IBM funded the research for at Berkeley on relational databases. Out of that project came Dr. Codd and all of those people who became well known in the relational database industry, including the guys who started Sybase and those other companies. They all came out of that project, System R. And eventually System R moved down to Santa Teresa and became DB2. My friend showed me this system and I thought, 'Wow that's really way cool. What's this all about? And I found out about Codd and Codd's papers and several papers on relational technology that had been published in *VLDB* and other places like the AFIPS proceedings. So I went and researched those and determined relational calculus was clearly the answer. I remember thinking, 'God, why didn't we have this earlier?'

Frana: Well, processing power was one of the problems, wasn't it?

Durbin: Exactly, exactly. So that's what we did. At that time we were building software products and so we built a layer that allowed us to have the programs operate as if they were relational and then we mapped those relations to traverse calls in IMS or IDMS or Adabas. And that's what we had to do until DB2. The program eventually ran on DB2. But DB2 wasn't out with production versions that we could use for about four or five years—not until about 1984—and I think the release in 1984 didn't support data types. It was really primitive. And so for years we used this interface layer that we had built to do the mapping. It was a fair amount of handcrafted work to do the mapping, but that was how we solved the abstraction problem. We wanted the developers who were working with these products to all be building things using the relational model. We wanted to structure and design our databases for relational, but we had to install on IMS.

Frana: Thank you for that because that is helpful to me. I interviewed Don Chamberlain and Ray Lorie and some of the System R people last fall. The story doesn't get told often enough that there was this sort of overlap period. Laypeople just assume that Oracle takes over and it's done with.

Durbin: In fact there's a question in there about IBM hegemony. I'll say it that way but one of the issues that really affected the software business a great deal was in the late 1970s—from about I'd say 1974 through 1982-'83, maybe even later to 1984—that IBM promoted IMS. I mean they sold IMS like crazy. And they sold IMS as a panacea. If you will just go put all your applications on IMS then magically you will have this database, you will have access to everything, you'll be able to analyze everything, every which way and you'll have control and yadda, yadda, yadda. It was a glorious vision but of course IMS by itself wasn't going to deliver that. IMS is just like anything else. You can do a terribly ugly IMS design just like you can do an ugly relational design or an ugly VSAM design. So the fact that you are on IMS as a database was not going to guarantee anything. But of course what was going to happen is you were going to get more memory on your machine, you were going to

pay IMS license fees, you were going to have to run bigger processors, more discs. IBM was happy as a clam because if you bought into the picture you were going to buy lots and lots of hardware and they saw themselves at that time as sellers of iron. They sold iron and software was a vehicle to sell iron and not much more. Of course when it became clear that DB2 was going to sell lots of iron because it used even more resources, they switched to that bandwagon. Of course, all those people who had invested gazillions in creating IMS databases were now spending the next five years, ten years converting those IMS databases to DB2 databases. I was thinking about that and at the time (1977-78) I was involved in doing a project for a company, a major personnel placement company. They wanted to make a decision between different databases and they didn't have criteria for doing that. Since I was known as a database hot shot they called me in. I said that we first have to understand what it is you're trying to do. So we looked at what they wanted to do. There was IMS, IDMS, Model 204 and Adabas. And I was predisposed to Adabas going in because Adabas came the closest to looking like a relational database at the time with its inverted file structure and I thought that was pretty cool. But when we got down to it, because of the kinds of applications they wanted to do, heavy transaction processing, it brought in several different paths. Network database was probably the best answer for them. But one of the things we did was attend a presentation that IBM did where they made this amazing statement that no other system, unless it's hierarchical, could be an effective database, which is of course nonsense. Five years later they were singing a completely different song. But they were taking the weakness of IMS and making it a feature—and I thought, what an incredible strategy. The weakest thing was that it only supports hierarchical, because you can implement hierarchical and network—no problem. You can implement hierarchical in an inverted system—no problem. But in IMS you could only implement hierarchical, you couldn't implement network. So you take the one weakness and you make it a feature because nobody else is going to say that we have that feature. That was really cool.

Frana: Tell me a little bit about some of the early companies you founded.

EARLY COMPANIES

Durbin: Let me sort of run through them. The first company that I co-founded was a company called Cybernetic Systems Inc., I don't know if that's on the list.

Frana: Cybernetic Development Inc.?

Durbin: No that's something different. Cybernetic Systems Incorporated was a consulting company. It was five technicians who got together and got contracts to build systems. We had a lot of expertise in real time online systems, and at the time there weren't too many people who were doing that. There weren't very many people who could build real time database-oriented systems in 1967 so we had some really good projects. That company lasted about three years and then the partners split up. Some of them went to various overseas location and so forth and so we dissolved the company.

Frana: This was Bay Area based?

Durbin: Yes, it was based in Berkeley. Then I started a sole proprietorship called the Institute for Cybernetic Development. This was 1970 and IBM had just unbundled its

services and one of the services that they had unbundled was education. We looked at the price of attending IBM classes and thought that there was a lot of money there so we put together a bunch of classes. We had a couple of people including me who were hot shot IBM OS systems types. We put these classes together and people loved it because we actually had people who knew what they were talking about. IBM would teach these classes with people who had been in a class on how to teach the class and if you asked a question they'd get back to you. We had people where you asked a question and they could dig right in and tell you why this was a good thing and that was a bad thing. We could immediately say, 'No, no, you don't want to do that because that component doesn't quite work right.'

Frana: So it could be a little more informal.

THE SECURE PRODUCT

Durbin: That's right. They knew exactly what they were talking about and that was a boom time for us. That was about 1970 to 1978. Then we founded Tesseract.

Frana: In 1979?

Durbin: Yes, 1979. Boole & Babbage bought the product Secure. Bruce Coleman, who bought it for Boole, is here at this conference. If you want another side of the story, you can ask him. That was in October 1978 if I recall right. Back in 1974, we'd been called in by a bank to take a look at some security issues. It was crystal clear that anybody who knew what they were doing could log onto the bank's timesharing system and crash that system in a minute. I mean give me a keyboard and I could have brought it down that quick. The systems were completely wide open because initially they were all under the control of the systems programmer. Nobody got near those things at first, but as soon as you started doing timesharing all kinds of people could get in and you could type in a scratch command and say, 'scratch the main operating system load libraries,' and that was it. When the system crashed, which it would fairly quickly, you wouldn't be able to boot it—you couldn't IPL it. And if they didn't have a backup IPL disc they were dead in the water big time. I mean I am talking about a $20 million machine that you could bring down with a couple of keystrokes.

Frana: Your demos were fairly effective, then?

Durbin: Oh yes. We'd tell people what was going to happen or what we could do. I said if you've got a disgruntled employee and you let him near a keyboard for five minutes you're toast. We built a system that stopped all that. Then IBM announced a comparable system. We were selling pretty good and then IBM announced a system called Rackout(?) and our sales went in the toilet because they didn't tell anybody what theirs did. We were out-marketed. They just said, 'We have a security system.' And it wasn't out for nine months. You couldn't get spec one on that system so nobody could do a competitive evaluation. Customers would say, 'Well, IBM has solved this problem.' But then when they finally got the specs of what IBM had done they said that this was unmanageable and our sales took off like crazy because they had broken the market open. They had created all this interest and now people could look at the two systems side by side and they would buy ours.

Frana: So the pre-announcement actually may have generated more sales for you?

Durbin: No, the pre-announcement would have killed us if we hadn't had the ability to survive on no sales. Fortunately, we had another part of the business that was making money. But we had no sales; I mean zip for six months. They just froze us.

Frana: And you think deliberately?

Durbin: They knew about us. We had gone to IBM. At that time they had a program where you could offer your stuff up and if they liked it they would resell it. So we had gone to them. But they said, 'No we have something competitive.' They had something in the works and they knew all about us but it was interesting. I mean they sort of did this on a regular basis—pre-announcing things just to freeze the market.

Frana: They have better controls on that today, like internal independent review.

Durbin: Well yes, and they've had a few consent decrees too.

TESSERACT

Frana: True enough. Now I came across an article that said that Tesseract actually ran schools like the Westin project did? Is there another Tesseract?

Durbin: No, that's another company with the same name. Tesseract started in 1979. When we sold the Secure package to Boole & Babbage we reinvested the money in an application software company. Then Tesseract became the lead company in doing real time human resource database applications.

Frana: Was this the first time you'd really thought about doing this, or had you been thinking about doing it for some time?

Durbin: Well, it came about because in 1974 we had built the prototype system for Bechtel as our first customer and when we got done with that project we said, 'We've got the only real-time database system.' And we went out to market that to the world, but we were just way too early for the market. People would go, 'Personnel? Why would we do this?' I was too early.

Frana: What was your argument when they said that?

Durbin: We had all kinds of reasons: return on investment, cost savings, better controls, access to information, and so on.

Frana: Were there new laws on the books that were making things more complicated?

Durbin: I don't think we had—oh yes, we did. There was EEO, an affirmative action program and certainly there was litigation support necessary, but it was hard for them to see why they couldn't get that out of a batch system. But in about 1978 IBM announced a product called Interpers, which was an interactive personnel system using IMS. And there was a pile of base code and a whole lot of assembly required. It was just a starter product,

but it helped and there we had a coattail effect because IBM announced this product. With Secure we'd had a coattail effect once people could look at the products side by side and then our product sales took off. So eventually there was a coattail effect but initially there was a freeze. In this case, IBM announced Interpers and we said that lightning could strike twice. There could be another coattail effect. And sure enough, IBM began to make the argument why you ought to have an online interactive personnel system. We came along with a vastly superior product and sales began to move. Tesseract became the leader and was acquired by Prudential in 1986.

Frana: How did you come up with that idea to swap service for equity?

FINANCING TESSERACT

Durbin: Financing a software company in 1979-'80 was quite a challenge. About that time we went out to the venture community and I literally heard from more than one VC, 'We don't fund software companies.' Can you imagine that now? Literally, we went to the venture community and they said, 'No, we don't do software.' There was a company called Matrix where a lot of VCs had put in a pile of money and come up short. So there had been a couple of famous failures and they were incredibly gun shy.

Frana: Was it the intangibility of the thing?

Durbin: Exactly. There wasn't an asset, they couldn't evaluate it, and they had no expertise. There had been companies that had made outrageous claims and cratered with absolutely nothing left. I don't even know what Matrix did—I think it was a service company. So they were really turned off and it was only a few years later, about 1982, when we found our first VC that would fund software. In the meantime we had to fund this company and so I did a round of financing with some private investors. Well, that only went so far because the deep pockets in that deal was a guy who had made his money renting construction scaffolding, a tangible asset. Every time we had a board meeting I'd have to explain what the business was and what the business model was. Eventually it got to the point where I said, 'I have to get somebody who knows what's going on.' Computer time was pretty expensive in those days and we rented computer time to test our systems. So I went to the vendor from whom we rented computer time and first negotiated a long payment schedule to help our cash flow. We negotiated 120-day payments on bills. Of course, by then we owed him a fair amount of money. So shortly after that I went back and said, 'How would you like to trade some of that debt for equity?' And I put a kicker in there. I said, 'Give us 24 months. We have a 24-month right to buy that equity back at double what you paid for it.' So the guy says that first of all it's funny money anyway because he's making money on his data center anyhow, so he's trading feathers for shells. He also knows because I've said this, that if he doesn't cut a deal we're likely not to be around anyway. So maybe he's turning bad into good. He has an upside potential and he gets rid of this big receivable and maybe it's getting a little soft anyway. So he did that deal. Sure enough, about 18 months later we did a deal with a VC and so we took the amount and doubled his money. It worked out. What it did for me is when I did the VC deal I squished the dilution out so now because the VCs paid substantially more than the double we gave him, it was reverse dilution for me to get more money from the VCs and take him out.

OTHER COMPANIES

Frana: And Cybernetic Development Inc., is that a separate company?

Durbin: No, I think that's the Institute for Cybernetic Development. I think it's mistyped in there.

Frana: I understand. What about Concur Technologies/Seeker Software?

Durbin: Well, Prudential bought Tesseract in 1986. That went along just fine for a while. We went from 55 employees to 180 employees in a little over a year. Great guns. And before long we had 50 percent of the Fortune 100 as customers. We're making good progress, but we're losing ground. Then there is a shift in technology. Meanwhile I was off and there was actually another company in between called Primrose. Primrose Software I think. And that was a spinout from Prudential to market some early stage technology for massively parallel computation. Well, we were too early in the market for that so we ultimately ended up using that as a tool for our client server stuff. Meanwhile, since I was off doing that, the company was losing ground to PeopleSoft and it was clear that Prudential was not going to step up with the investment needed to build a whole new version of the system on a client server platform. So I went and found an investor group that would take Prudential out and put some money in the company and we could go take a run at PeopleSoft.

Frana: This is when you got your parallel processing patent?

Durbin: That was Primrose. That was about 1989, I think. Prudential funded a research project that produced the parallel processing software and Primrose was the company formed to see if we could market that. The timing was a bit off because we were starting into the market just when the massively parallel computing companies were going down the tubes. They hadn't committed. If we'd been a year earlier and they had had more vision we might have been able to put something together. The problem with the massively parallel companies was that they were all competing for this little bitty scientific market. The huge market was the commercial market and what we had was a piece of software that would allow commercial applications to use massively parallel systems. These hardware companies were going down the tubes just about the time we were ready to go to market and so the timing was really off. So I put together the investor group to buy Tesseract back from Prudential. Eighteen months later, Ceridian bought the company because they were the leading payroll processing company, second to ADP, and they needed Tesseract's technology to build a new payroll system. I stuck around for a little while and then left to form Seeker Software at the end of 1995. Seeker Software was one of the first Web based application companies targeting delivery of applications to employees and managers within the Internet.

Frana: Like a portal?

SEEKER SOFTWARE

Durbin: Well, we were doing the transaction side. Portals typically do the information organization side. We did transactions like, 'you want to move an employee from this

department to another,' or 'you want to hire somebody or give somebody a raise,' or 'put somebody on a project, or 'pay time, travel and expense vouchers'—those kinds of applications. Things that managers do everyday and you ought to be able to do through Web applications. That's what Seeker did. Concur bought that product in 1999.

Frana: In a PR Newswire story about Seeker one of the things that you said was that the Web offers tremendous advantages over self-service approaches that follow serial logic whereas the human brain can assimilate and act on information delivered simultaneously from several sources. I mean that's the kind of thing that Tim Berners-Lee has always been talking about—any other approach just doesn't match the way people think.

Durbin: I agree completely. One of the things that we tried to do going into Seeker was to say, we are going to use the Web metaphor. We're not simply going to deliver an application over the Web meaning IE and browser. We're going to use the Web metaphor. Part of that means that people will be able to navigate to things when they don't quite know where they're going because that is the way the Web works. I mean you get someplace and you go, 'Oh wow, and you click there and you eventually get to the place. You don't know how the hell you got there and in fact you can't get back. That's the nature of it. In most applications the user interface is very highly structured: 'We'll do this, we'll do this, we'll do this, we'll do this.' Well, that works just fine if you've done the operations research behind the data entry organization in a back office and these are the forms they are using. But if you are dealing with novices, and they haven't been taught to enter this or enter that—you have to have a completely different model. Much of the work to move applications to the Web involves taking those back office applications that were highly structured in their process and move them to people who have no idea what process it was this thing was implementing.

Frana: So they don't really need to see all of that.

Durbin: That's right. You need to provide bite sized pieces and lots of links that allow people to go where they want to go, and when they get to something they can go, 'Oh, that's not right,' and there's a button that says, 'update this.'

Frana: So it's intuitive.

Durbin: Exactly. It has to be. It has to be something that people who don't have Web skills will be able to use. What I said was that we wanted an application so that if you couldn't find Mickey Mouse on the Web, but you could find the Disney Web site for Mickey Mouse, you could use this application. And oh by the way, if you couldn't do it you could have your ten year old show you how to do it.

Frana: Who were your competitors?

Durbin: Well, most of the competitors ended up self-destructing in one way or another. Part of the reason was we had strong technology. We did not come from some other technology. Now Edify came from voice systems and they had an application that took their voice technology and put that stuff on the Web. Well, talk about structure! You'd type a character and then it gives you data and you type in a character and so on. This was a voice system. Because you have no visual input—an auditory system is linear—it has to be

structured that way. So they took a framework for that kind of linear structure and put it on the Web. It was completely unsuccessful. It was not difficult to compete with them. But they had a big marketing force, big sales force. PeopleSoft from time to time was a competitor. Who were the other competitors? A couple of them just came and went inside of a year or so. Our biggest concern was the ERP vendors, but they have such long development cycles and old technology that we were able to be several years ahead of them.

MARKETING SEEKER

Frana: There are a lot of software anecdotes in the Volume 24:1 issue of *IEEE Annals of Computer History* from various people who started companies. One of the things I noticed in these accounts was that each person had a 'gimmick,' a sales strategy that seemed to work very well for them—whether it was borrowing from future sales and bringing them into the present or reference sales or special pricing strategies. Did you have a 'gimmick,' something that worked effectively but was sort of unexpected?

Durbin: Yes, the key thing we did with Seeker was a demo. We put the system on our Web site and you could go there and run a demo of the application. That helped develop our credibility. Tesseract was all about real time. The pitch was having data immediately accessible. We would have anecdotes—like the CEO of Bechtel calls down and says, 'Do we have anybody in San Francisco who speaks Russian? I've got some Russian visitors coming. Tell me if we have a civil engineer who speaks Russian.' And the answer was, 'Yes, we have three of them.' 'Give them a call and get them up here.' As a consequence, Bechtel got a couple billion dollars worth of business because there was an engineer there who spoke Russian. Talk about making the pitch for having accurate and timely personnel information. There it is. We moved it beyond reducing headcount and making it a more strategic thing. We did big pitches at Tesseract about real time and later that shifted to functionality because we had the functionally deepest product. We could ask the questions: Does the other product compute this or that? Do they have payrolls that look like this and this and this? Can it handle benefits that look like this and this and this?

Frana: You had a checklist.

Durbin: Exactly. You create the agenda and the other guys can't beat it, can't check all the boxes and get the business. The strategies change depending on where you are in the cycle of the company. Early on it's got to be about a vision of some kind and later on it typically shifts to functionality, robustness, durability, performance—those kinds of things. I'm sure you're familiar with Geoffrey Moore's book *Crossing the Chasm.*[2] When you are selling to guys on the other side of the chasm it's all about functionality, performance, durability, reliability, and adaptability. So the slides for the same piece of software change dramatically from the early stage where you're selling the vision and generally the vision in the early days was self-discovery, management applications which would automate the whole company without having to train anybody because it's the Web after all.

---

[2] Geoffrey A. Moore, *Crossing the Chasm: Marketing and Selling Technology Products to Mainstream Customers* (New York: HarperBusiness, 1991).

WAS IT VAPORWARE?

Frana: Did you ever get in the position where you felt like you were selling vaporware and then you had to sort of catch up?

Durbin: Oh yes, we laughed about it in the industry. We'd say, well you have napkin-ware, slide-ware, and then you get software. Somewhere along the line after you're doing the slides and the promotions you really do have software.

Frana: Because this is just the development cycle, is that what you're getting at?

Durbin: Part of it is the development cycle. Part of it is a marketing thing to grope and find the market. In the case of Seeker we built prototypes of things and then we would see whether or not those things resonated with the customer. When we got to actually do a deal we were very clear with customers where things were with this piece of software. We told them there are five things you want: these three things are fully tested, these two things are in development, and we will commit to deliver these things at this point in time according to this set of specs. So we never sold vapor. It's really about putting together a picture of the vision and finding out what parts of that picture resonate with the marketplace. And then there is an integrity issue. My companies did not sell things that didn't exist. We were very clear. We'd get the customers and we would make it very clear at some point in the field exactly what was and wasn't, but not necessarily when we did the demo. Our sales people didn't stand up there and say that this is only a prototype—they didn't do that. We needed to win the business, but before we got to where anybody's career was on the line about what we were going to deliver, it was crystal clear. We would show them our product plans. We would show them where we were. They knew going in what we had and what we didn't. There is a legal requirement there. I mean if you do too much of that you're going to be in defraud-land. But even more importantly, you have to develop an integrity relationship with your customers and you have to be clear. They will accept that if you go and say, this is what we have, this is what we don't have and this is what we're doing. And you're clear. That's cool. But if you lie, they are going to hate you even if you deliver the best stuff. If you lied about it, they are going to hate you. So you can't do that and my company always had a reputation for integrity.

Frana: I wasn't questioning that obviously.

Durbin: Well, it is an issue in this industry and I understand it. We competed with companies who were selling vapor and were not as straight about it as we were and sometimes we lost business and the customer would go down the road. In some cases they would come back to us and say, 'We went down the road with them and they don't have it. They said they had it but they didn't.' We'd get them back again, having thought we lost the business. Technically, we lost it but we'd get it back again. So, it is something that happens.

USER GROUPS

Frana: Did you have some strong user groups that helped build trust in your relationships?

Durbin: Actually, I have a story about that. I went to an ADAPSO meeting in what must have been 1981 or 1982. Tesseract was underway. We had a couple of customers. There was a list of ADAPSO sessions and one of them was on user groups. I didn't know much about how they worked. This was the most amazing thing, and you've undoubtedly heard Luanne say that this was what was magic about ADAPSO: you would get three companies who in ordinary circumstances were competitors and would cut each other's throats, but when we came to ADAPSO somehow we could suspend all that. So here were three companies—two of them were actually competitors and one of them was in another sector—and they got up and talked about their user group and how they had organized it and they had three different models. One of them owned the user group. They had somebody on their staff running the group and they charged people to attend. It was their thing. Another one had let the users run it and the third one had something in between. I listened and each one gave the pros and cons of their approach. It was just amazing. I came out of there having learned so much that I can't imagine how I could have gotten that perspective except through trial and error. I would have made serious bumbles and mistakes along the way. As it was, I came out of there, went back home, talked to my management team about what I had learned, what the pros and cons were, and we discussed it and decided how we were going to go forward. For at both Tesseract and Seeker the user groups were fabulous resources. We used independent non-profits that were set up with officers elected by the customers. The customers had meetings that they invited us to or in some cases, they didn't. It was a living organization on its own. Now there were times when that was a problem because we would have some problems with the release of something and they would criticize us, but what it meant was during the early sales process we had incredible references, customers who believed in the company. They were *part* of the products. People would call them for references and go, 'Wow!' So from that point of view it was absolutely the right thing to do. But it was because of ADAPSO that that happened.

ADAPSO

Frana: We should talk about ADAPSO. What else do you think was magical about it? Was it that you all felt like you were in the same boat? There were certain pressures out there on all of you that you shared—like IBM.

Durbin: Yes, it was a new industry. In part, we had real conflicts with IBM's hegemony. IBM's control of the commercial computer business in the 1960s and1970s was very strong. If you wanted to do anything you had to think about what IBM was up to and if IBM wouldn't like you doing it. They were pretty cavalier about their treatment of independent software companies. If you understood that they were there to sell iron then you could have a relationship with IBM. Where you had very low expectations you could operate perfectly. They were really not anybody's friends and in many cases because of their own interests I think held the industry back. We had this common ogre, sort of like Microsoft today. That brought us together. Also it was an early industry. So many of us were groping our way. You know, Larry Welke published the first directory of programs.

Frana: The *ICP Quarterly.*

Durbin: Yes, I mean who would have thought what a resource it would be that there was someplace that people could go and they could find the name of your company. Marketing was a real problem. We had no idea how to market because there weren't developed strategies on marketing for software companies. It was, how do you do this? So a lot of us, so many of us, were groping our way. It was a young industry. I think that made a big difference.

EX-IBMers

Frana: Were there a lot of ex-IBMers that became ADAPSO members?

Durbin: A whole lot of people who came into the software industry were ex-IBMers. A lot of us had an informal rule that you could hire ex-IBMers only on the second bounce.

Frana: Because they were bitter?

Durbin: No, because they were used to the big corporate environment. You know, they could pick up the phone and 'things happen.' And then they go into a software company, and here's twelve people and they pick up the phone and they're talking to themselves. So they needed to go through the shell shock of working for a non-resource loaded company. IBM was so profitable in the 1970s; they just had bags of cash.

Frana: Now, they couldn't spend money like water.

Durbin: An executive at IBM, a sales guy in IBM, I mean he had the whole company at his beck and call. All he had to do was say, 'I think we can place a new machine here,' and people would just come out of the woodwork. Technicians and experts from around the world would fly in. It doesn't happen like that in small software companies.

Frana: Yet you saw a lot of people leave IBM. What kind of person would leave that utopia?

Durbin: Well, certainly in the 1970s not too many people. It wasn't really until the 1980s and later that you began to get people in the HR business. Duffield came out of IBM. He'd been a salesman in the Northeastern region. No he was a technician, a systems engineer. It was his brother who was in sales. That was 1978-'79 when he did that and he was one of the few. Most of the people in the application software business were not from IBM. Then beginning about the mid-1980s we began to get people from IBM. By the end of the 1970s there are a lot of people becoming disillusioned with IBM.

Frana: The culture?

Durbin: The culture, their hegemony was beginning to slip. There had been a couple of antitrust suits that had clipped their wings a bit. They weren't quite as strong.

ADAPSO ACCOUNTING COMMITTEE

Frana: Which ADAPSO committees were you on?

Durbin: I was only on the financial accounting committee. I didn't participate much in the committees. I was CEO of a start-up and I didn't have lots of time. I benefited greatly by my ADAPSO experience but I didn't give much back at the time.

Frana: And why were you interested in the financial accounting standards committee?

Durbin: Well, it was fairly personal. We were a software company. We had invested fairly heavily in our products. This was back in the early period when we couldn't get VCs interested. So part of what we had to do was finance the company through banks, and we wanted an asset on our balance sheet. If I went out and built a building for a million dollars I'd have a million dollar asset on my balance sheet, but if I went out and built a piece of software for a million dollars I had nothing on my balance sheet. It didn't make a lot of sense. That was my argument at the time and in fact we found an accounting firm, Deloitte & Touche, and made that pitch. They agreed with us and so we had audited or at least reviewed balance sheets to show the bank. And then everything started shifting around. Deloitte & Touche said, 'We're not sure this is such a good idea anymore.' There hadn't been a declaration from FASB yet, so the accounting firm said, 'Well, we said it was good last year, but this year we don't think it is a good idea. So we're going to take this off your balance sheet.' I said, 'Whoa! Time out! You can't do that!'

Frana: Someone last night said that the software product people in ADAPSO were kind of States' rights types and everybody else—including the data processing services people—were federalists. That is a very curious comment. Were the software people much more in favor of local control as far as laws and regulations?

Durbin: Well, I think we were. The software companies were smaller compared to ADP. Even today there's hardly a software company that is that big, except for Computer Associates, Wang, and Microsoft.

Frana: But they were really the five hundred pound gorillas.

Durbin: Yes, so they were big and we were different. We had very different interests. It made a big difference to us about having things like revenue recognition standards and software capitalization standards and so forth and these guys didn't care about that at all. It was just fine to have to write off all costs of software development. I think it was much more that the software companies were scrappy; you could call it stage fright, you could just call it scrappy. We had a different set of interests and we looked at software in a very different way than these guys did. These guys considered it a pain in the neck and they'd just as soon try to figure out how to do business without it, but it was our lifeblood.

SOFTWARE INDUSTRY ASSOCIATION

Frana: There was also an independent software industry association?

Durbin: SIA.

Frana: That was Welke's group?

Durbin: Yes, but it was formed within ADAPSO.

Frana: It was?

Durbin: Yes. Initially it was just a special interest group within ADAPSO. Then eventually it split off on its own and became much more independent because of the 400 pound gorillas that were in the same room and SIA couldn't get its interests promoted. I was at one of the early meetings where it was the software special interest committee or something like that and it came up with a whole bunch of things. For example one of the things software companies wanted—and this was a big issue—was some kind of protection. IBM didn't. In fact, there was a big issue that when the PC came out the software industry wanted a mechanism so that we could license software to a specific PC and we could have a lock and key mechanism for each PC. IBM would absolutely not do that because they were happy as a clam if software was pirated like crazy because people would go buy more machines. As far as they were concerned they were in the hardware business and they didn't give a damn if the software that was running on it was ever paid a fee. So they were happy and just fought licensing and the mechanisms that we invented in the early period so that you couldn't use the software unless the disc was in the machine.

Frana: Anti-piracy.

Durbin: Yes, the anti-piracy stuff. Well the whole anti-piracy thing was in fact one of the things that caused the software industry group to split because these other guys had no interest in that. In fact, IBM fought software licensing like crazy until about 1984 when somebody at IBM realized that IBM was getting a vast amount of high margin revenue from software licenses, and maybe things like DB2 and so forth were really good ideas and maybe they ought to switch their position. They spun around so fast you could hear their neck snap, and came over to the other side. But in the early period we had to fight them like crazy. But early on, particularly on the PC stuff like the early versions of Lotus and VisiCalc that incorporated things where you had to have the disc in the machine—they hated that.

Frana: Why is ADAPSO transformed into the ITAA? What happened?

Durbin: I don't know. I sort of dropped out of ADAPSO about 1990, which is about the time that it was headed down. I think part of it was that the industry became more mature. In the early days there were so many lifeblood issues like the software accounting issues and the software protection issues and the license agreement. The first license agreement I got for software was from ADAPSO. It was a model agreement for licensing.

Frana: You got the forms from them?

Durbin: Yes. That stuff helped because there were so many fronts on which you needed any help that you could get as a software entrepreneur, like the user group issue. I ended up spending so much less time because I attended this couple-hour session and brought that stuff back and we moved forward. It was like that with so many areas. Now it's a much more mature industry. You can hire people who have been in software companies for years. At the time you couldn't hire anybody who'd been in a software company for five years because the

average software company was two years old. So I think in part it has matured and therefore its needs have changed. We had lots and lots of issues to fight and we had the one big IBM to deal with and now we have Microsoft.

Frana: Did you ever use the contracts directory?

Durbin: Yes, there was a book published.

Frana: That was a lifesaver to you?

Durbin: Absolutely.

Frana: Were you in the organization early enough to remember all the problems with the banking industry and the controversy there, whether they were violating the law?

Durbin: I'm not sure. What were the issues?

Frana: You came out of that culture in a way, which is why I ask. Banks were creating these systems and ADAPSO really wanted them to get out of the software business, to divest themselves because that wasn't their business and they were making an awful lot of money and competing very hard against some of these independent software companies.

Durbin: Service companies too. Service companies were getting uptight about the banks doing payroll services and essentially giving it away to get the balances. That was actually more of a services-side issue than it was a software issue. Banks were never giving away software. It was the services guys who were really upset about the banks. That's why it wasn't a big thing for me. ADP and so forth were going to fight that battle.

UNBUNDLING BY IBM

Frana: I know you've said a little bit about this, but did 'unbundling' affect you directly or was it one of those things that only in retrospect really seemed to make an impact? Some people talk to me about it as if they heard about it, but it didn't change their lives immediately. It only did later.

Durbin: It was a big thing for us in a couple of ways. First of all the company that I started at that time made a lot of money because IBM had set prices for those classes and we were able to give them vastly better classes for the same price—so unbundling was great on the education side. I think it was also pretty significant on the professional services side. In the 1960s, if you bought a piece of hardware you got a systems engineer—you got people for free to help you build applications. After unbundling you no longer got those people for free from IBM. Essentially those were technology and knowledge transfer people because IBM was working on some project at one bank and then they would go do it again at another bank. If you think about it, that's a whole pile of intellectual property to transfer over. If you bought a machine what you also ended up getting was this intellectual property. When that was no longer the case, then it made the real cost of building software for the bank a lot more clear because you no longer got a hoard of IBMers in there to help you do this. They were interested in selling the clicks on the machine so they gave the people away. But now

the cost of software became much more clear and I think you got people a lot more interested in application products because if you didn't get 30 SE's for free and you had to go hire those people and train them and go through all that stuff—the software development was a lot more expensive.

Frana: That makes a lot of sense because this is when application software really takes off.

Durbin: That's right—1970 was really a watershed time and I don't think that the impact was quite what IBM wanted. I did learn one thing from them though that I thought was really cool. In the mid 1960s I was a software manager of subsystem software for Pacific Intermountain Express and we had this SE from IBM. I mean the guy was free, he would come in and help people and so on. But then after awhile I figured out what was going on. I was thinking on the one hand, why do we get this guy for free, that's really cool but what's going on? And then something would happen and 'Bam!' the IBM salesman would be there. I thought, 'Ah ha, of course, he's intelligent.' If anything twitches, the SE knows what's going on and he feeds it back. He's an on-site spy. What an amazing concept. Several years later when I had my first major account, which was doing personnel services for Bechtel, I put an SE on site. She had an office, a desk right outside the VP of HR's office. Anytime anybody from IT walked into his office she knew about it and I'd be over there the next day and if any one of his people was unhappy, she knew about it. She knew where they were going. She knew what was happening. I mean it was like having my finger on his pulse, day by day it worked so well.

WOMEN IN SOFTWARE

Frana: What was it like for women in ADAPSO and in the software industry in the 1960s and 1970s? I know just a few of them like Luanne Johnson and Barbara Brizdle and some others. Was it tough for them?

Durbin: You know, it's funny—I've thought about that a lot. Tesseract was pretty *avant-garde*. We were in San Francisco and the monthly meetings or quarterly meetings of the Women in Computing Society started at our office among some of our people. At several points in Tesseract's life the company had a majority of women. Seeker had a majority of women until I called my managers in one day and said the next hire has got to be a man. I want some balance here. In the 1960s IBM hired and trained lots and lots of women as systems engineers. Now they had almost a two-tiered system going. The managers were all men but the people who worked with the customers—a great many of those were women. Some of the most senior SE's in IBM were women and they moved up and I think IBM had a pretty good program for providing opportunities. It didn't move into the executive ranks, but as systems engineers—people who really knew about software—I think IBM did a really good job. Their recruiting people would go out to campuses and they would recruit men and women. In fact, later on when somebody said something about discrimination I said, 'Well, I don't know.' All the SE's I ever met were women. Not in this industry. And it's spotty. Probably much less on the West Coast. But going into the software business it didn't quite work that way. You know we certainly recruited some women. I had early on—in the mid 1970s, we had women executives in the company. But for some reason the women that went into IBM and became SE's didn't leave to go into the software industry very much. They

sort of stuck with IBM or they went off and got married and left the industry altogether. Most of the people who came into the software industry from IBM were men and I think it was because in part there was a glass ceiling at IBM. We would hire ex-IBM executives but you couldn't find any women ex-IBM executives; they may have left because of the glass ceiling.

REPLACEMENT OF PEOPLE

Frana: The other question that we dance around a lot is the 'unintended consequences' question regarding spillover costs and benefits from the software industry. How did you respond to the people—and the literature is replete with this, especially in the 1960s and early 1970s—who saw software as a cause of technological unemployment: people presumably lost their jobs to automation because of the software.

Durbin: Yes, there was a quandary about that, particularly in the 1970s. In fact a question my wife asked when we first met was, 'So, what does your software do? Won't it put people out of work?' The way I answered that was to say that most of the jobs we were automating were not jobs that people enjoyed: sitting down and keying data is not a particularly fulfilling job. I didn't have an explanation about what it was that people were going to do that was more fulfilling, but we certainly knew that many of the jobs that we were automating out of existence in the 1970s were jobs that were fairly low level. Now, of course, you could argue that we are improving the productivity of the organization, and if you don't do that the organization won't survive anyway. So those companies that didn't automate are gone. It was really a competitive issue, as it certainly is now. But the thing that I tried to do was to make software that would be something that people could engage with.

Frana: Empowering?

Durbin: Exactly. Empowering, engaging, so that the software would improve the experience of the people that we interacted with; that became our watchword both at Tesseract where we created a very powerful, adaptable systems and at Seeker where we were dealing with systems that were designed for users in the workplace. What we were trying to do was to make people's work more engaging and yes, we were going to improve processes. Getting rid of some of the folks who are pushing paper around and hopefully moving them to more productive work was going to benefit the company as well.

Frana: Well that argument seems to have won. I don't hear people talking about software costing jobs anymore. They talk about it in terms of job creation.

Durbin: Exactly, but I think that has to do with the attitude that we weren't simply trying to do the operations replacement mechanically. There was a quality to the software of improving the experience that people have in interacting with it that made a big difference. That stuff came about as user interfaces got much better with things like Windows.

STOCK OPTIONS

Frana: Speaking of your employees. There is an article by the historian Tom Haigh in the new *Annals of the History of Computing* where he talks about stock options being offered to

skilled programmers as early as the mid 1970s. Is it possible that programmers could get into these companies that early with options?

Durbin: Yes. In every company that I've had—beginning with the one in the 1970s—we pushed stock options all the way down. Certainly to all the technicians because that was a way to try to hold onto them and make them part of the picture; but even down to the administrative people.

Frana: Was all that because of Black/Scholes? I don't know how much of the option pricing model you've studied, but it's the pricing of those options that somehow was a stumbling block before.

Durbin: Right, it was the change in taxes that deferred the realization of value way out into the future. It's hard to describe it. I was going to use the word fantasy. That's not quite right, but it was this vision of everybody being part of the entrepreneurial experience. Because in many cases, software companies in their early stages were pretty rugged things. I remember the programmer for Secure. Because we were getting into the product side, we finally had to actually get an office. Before that the company had sort of been a floating craps game. I had a little office in my house, but I would go and meet with people that were working for us at the project site that they were actually working on. We'd have an "office" in their cafeteria.

The company didn't really have a central office, but when we got more serious in the product business we really needed an office. We got some space and told the programmers working on that product, 'Okay, you have to go there.' But we neglected to take into account that we also needed a desk and a chair. So you'd see this guy sitting cross-legged on the floor with a terminal. Here's a 500 square foot space and one guy sitting in the middle of the floor with a terminal and that was it. And then I said, 'Oh, you have to get some office furniture.' I was talking about this with my customer at Bechtel and the guy said, 'You know, we have a whole bunch of office equipment sitting in a warehouse.' And I said, 'I'll tell you what. You can store it at my place. I won't charge you any storage.' And he said, 'Sure, that's fine.' So we went over and picked out a couple dozen desks and a bunch of chairs and so forth and he stored them at my place for a year or two so I didn't have to buy furniture.

To build software in those days in particular we had to buy this really expensive computer time. Building software is expensive. It's really, really costly. And that was part of the issue about capitalization of software. If you're just building systems for people that was expensive, but building quality products is like ten times more expensive because of the requirement for packaging and the documentation and the quality assurance and the adaptability that you have to build in and so forth. If you are building software that costs ten times what it would cost to build a one-off and the cost of that is all people. You put all your money into people and these people had to feel like they were part of the picture. Even at Seeker Software, which in 1996 opened its first office, we got dirt-cheap space and the air conditioning would fail all the time and we had people sitting around drenched in sweat, writing the software. Well, they had to feel they were part of something that was going to happen so they were all loaded up with stock options.

Frana: Were there other industries that really took advantage of that? I mean, it makes sense, it seems so logical, putting the two things together, what your needs were and what the employees wanted.

Durbin: We were serious. There was a bit of idealism even on the part of executives that we were going to share the wealth. That was part of it too.

EMPLOYEE ATTITUDES

Frana: This wasn't going to be your traditional hierarchical organization?

Durbin: Yes. This wasn't: 'If you work for me, this is the way it is, you get your paycheck and you do what I tell you.' Because these are not those kind of people. You are talking about mavericks, people who are crazy. I remember walking in one morning and I walked around and here was one of my programmers and she was working at the terminal and I walked up and said, 'Wow, you're here early.' And she says, 'No. I got to working on this problem last night and I looked up and it was 3 o'clock in the morning and I said, well hell, I'll just finish it.' Where do you find people like that? But those kinds of people are exactly what you are looking for when you interview these people. They are just a little strange.

Frana: They don't think inside the box.

Durbin: Exactly. So you want those people and they don't work for paychecks. They work for visions and being part of something and that means equity too.

Frana: How did you want to change the world and how did your employees want to change the world? What did they talk about? What was the vision?

Durbin: Well, one thing we had at Seeker was something we called the 'Culture Club.' This was a very ad hoc meeting of anybody who wanted to come. There would be an email that would go out saying that the next meeting of the Culture Club would be on Thursday at six o'clock. People would show up. I would show up from time to time but I didn't go all the time. What the meetings were all about was, 'What do we want to be when we grow up? What kind of values do we want to infuse in this company in order to make it something we are really proud of.' They would come up with terms like 'integrity' and then they would have a meeting and just discuss 'what does that mean?' 'What does it mean to say we want to have integrity or we want to be entrepreneurial?' What does it mean? And that's what the meeting would be all about—we're going to talk about integrity and people would come. They would come out of there usually with some kind of thing. I remember the one on integrity because I thought that was particularly cool, and it came from one of our programmers. He said that integrity is being who you say you are; it's just that simple. If you say you're this you better be it and the other way around. If you're this, you've got to tell them what you are. That's what integrity is.

Frana: And they really meant this?

Durbin: Absolutely.

Frana: It wasn't a marketing ploy?

Durbin: No sir. There were no marketing people in these meetings hardly. Occasionally the product people would walk in and sometimes the VP of marketing might go in there, but we were not creating spin. We were creating what it was we wanted to be. Well, that was one kind of thing. We tried other things that didn't work. I brought a guy from a Buddhist monastery in to do meditation classes, but we did these meditation classes at like 6 o'clock. Well, by 6 o'clock everybody in that company is so wired this guy would come in and these people were floating off of the floor. It didn't work at all, although it was a great idea.

INDUSTRY VISIONARIES

Frana: Whom do you admire then? Who exemplifies these kinds of ideals?

Durbin: Well I think Steve Jobs has some of that, although his interpersonal stuff is legend.

Frana: It leaves something to be desired.

Durbin: Yes. He certainly has the vision side of things and people will buy into his visions, but then there is sort of another side of that. And in many ways the one that I would credit with being the most strong-headed is Wosniak who actually had the other side of that.

Frana: The technical side.

Durbin: The technical side, but also some of the vision side. He didn't want to be somebody who kicked people in the teeth to make them buy it. Who else?

Frana: I'm not just talking about popular people.

Durbin: Chris Date. There's a guy that clearly has the vision of the abstraction and was able to take the obscure stuff that Dr. Codd had done and transform it into the real world and was a really nice guy besides.

Frana: He was a great popularizer for relational technology. Is he still consulting? Is that what he does now?

Durbin: I don't know. There was a Date-Codd Institute a number of years ago. I don't know if that's still alive. Let me chew on that question.

Frana: More pragmatically, a lot of people just down the street here in Congress are worried about losing our lead in software to Indian programmers and people from abroad. We are not training our native resources very well.

COMPUTER SCIENCE EDUCATION

Durbin: Oh, there was one committee I worked with. And that was the education committee.

Frana: Is this the kind of issue that they might have discussed?

Durbin: Well, they did this incredible piece of research in 1986-87 that showed that the computer science curriculums were graduating 50,000 people fewer per year than the industry needed.

Frana: When was this?

Durbin: The mid 1980s. You fast-forward ten years and we're about a half a million people short. Duh. So there it was, there was all the research and all the data and so they asked a number of us if we would give speeches and so forth and they put a videotape together which some of us distributed to high schools to try to get people interested. 'Hey, I'm a software executive I want to come and rap to some of your people.' We needed a hell of a lot more administrative support for people to go. If somebody had called and asked me to make an appointment to go speak, I would have spoken, because I did a lot of speaking and I was comfortable doing that. But to actually have to do all the grunt work in order to get in to do the speaking, I didn't do very much of that because it was like 'let's see, how much time do I have to do this.' It could take two days to get a meeting set up and school bureaucrats are notorious. They want to know if you've ever molested kids, it's who are you? So that program didn't work well and obviously we never got the interest to fill that hole.

But I tried to work just two years ago on a program. We could solve this problem real easy and go train people here. If you are an underprivileged kid in Oakland, you can't go and get the kind of training that I got when I first got interested. Just provide basic technical training, do Windows network administration, just some basic classes, get that stuff going and get our people trained here. The only reason that we would be losing to India is because they've been trained. Go look at their school system and then go look at our school system. And then wonder why there is there is a gap. Give me a break. It's obvious. So step up, provide money for education and we'll solve the problem. If you don't pay for education you're not going to get it. Look at what the Indians spend on education per capita versus what we spend.

Frana: I'm reading through lots of memoirs. I noticed that a lot of people have non-traditional backgrounds. They come with all sorts of different experiences.

Durbin: Well, a lot of us are dropouts. I'm a drop out. I dropped out for a while and I got into programming and then I said, 'Well, okay, now I made some money, I'm going to go back and finish my degree.' Guess what. When I went back to school, I could teach all the classes in the computer science group. I mean I literally could teach them. There was one I couldn't teach. And that was a class in analog computers. I'd never had my hands on an analog computer. So I took a class on analog computing and eventually I said, this is pointless. I'd talk to people. I attended a few classes because it was easy A's. But I'd say, what are you going to do? Well, when I grow up I want to be a programmer. So I ended up just dropping out as a lot of other people did because the academic programs were not on the mark. They are typically behind the industry and part of it is that the salaries of programmers are twice the salaries of instructors.

Frana: So, a lot of on-the-job training goes on these days because there is such a disconnect between the academic and the professional?

Durbin: Well, no, not as much. In the 1970s at ICD we were short of people, scrambling like anybody else. So I said, 'Okay, we're going to start training some people. We'll have a quota; we'll do three people.' But because we had done training earlier, we were now mainly doing consulting work—but we had done training so we had all the course stuff. There were a couple of us around that could teach and so we would bring in hired people and the deal was, you go to work for us and we'll have Tuesdays and Thursdays from 6:00 to 10:00 PM or something and we'll have one of our people come in and we'll take you through this training program. So these people had to work long hours, they had to work plus they had to attend classes and so they buy into it. Bt what we discovered was, they'd go to work for us and we'd take them through the training program and then somebody else would offer them damn near twice what we were paying them. So we stopped it. I had to stop it because I couldn't break even. And a lot of other people shut down their training programs because as soon as you got somebody out of training, somebody would buy them off. So I don't think there is nearly the same kind of training in the industry today that there was twenty years ago, by any means, because the demand is so high. In 1997 and 1998 when we were recruiting like crazy at Seeker we'd hire a programmer if he could fog a mirror. The demand was so great who could do a training program? You'd be crazy to run a training program. For years training programs have been unprofitable. It used to be that places like banks and insurance companies ran regular training programs because they would find people internally and they would test them and they would run them through the programs. Then the software companies got hip to that. Who graduated from a class this week?

Frana: So what do we need to do differently to close the gap?

Durbin: In part, I think we need to move way back into the high schools so that people understand that it is not magic, that they don't have to have four years of calculus. What they need to have is certain kinds of conceptual thinking. We need to teach conceptual thinking and we need to introduce people to computers and the whole range of things you can do from network administration to systems design and teach people what it is all about. We used to do that with automobiles. We used to have auto shops and radio shops and wood shops and stuff like that. I was on the board of a private school and we had a computer lab. The school had a $20 million endowment, so we had all the computers we needed. We had a teacher who would work with students and there was no senior who didn't have a Web page, who didn't have a whole pile of skills. They thought about it just automatically. We need to push that way down. It's not the college curriculums that don't address the needs of the industry. We need to get people on track earlier, much earlier.

Frana: Anything to be said for the distance that seems to be growing between the machine, the programming, and the end user?

Durbin: A friend of mine died last week and I was at his funeral. He was a great programmer and there were some other people there. One of the guys I drove out with was another great programmer and we were talking about that back in the 1960s and the 1970s we actually saw the machines that we were working on. By the time you got to the 1980s, most programmers would program and had never seen the machine their code was running

on. They would submit stuff and they would test back and forth but they had never actually been inside a computer room. And if you stand around in a computer room long enough you learn that your feet freeze and if you haven't had some of those kinds of physical experiences you don't know the nature of it. The PC has sort of that experience since people can now actually physically see the machine but very few people are able to get down low enough in the operating system to get the structural feel of it. I think education could solve that a lot. There's a whole variety of ways that probably only college level courses could help people dig into things so that they could understand what is really going on underneath the covers. There is a problem in dealing with it when everything is at such a high level of abstraction.

Frana: I'm wondering if we did too good a job hiding things from people so that they could use them?

Durbin: Actually, we were talking about that and our conclusion was it's sort of a misdirection. The objective was not really to hide things from people, the objective was to create components. If you've ever written a date routine, you really don't want to do that again. What you'd really like to have is a function, an object so that you can give it a date and then ask all kinds of questions about it. That's probably a great case—understanding what would go on inside one of those components—would be a great exercise for a sophomore level computer science class. I never looked at the curriculum.

Frana: I'd be surprised if that gets taught.

Durbin: Yes, but that is really all you have to know about component software. Once you understand what component software is you can begin to understand why when you are talking about something it's a 'component.' And that's what this is all about. The thing of it is that you don't have to cover the whole ground to get the ideas as to why you do component software in the first place. It's because once you've coded this stuff it's really boring to do it two or three or four times.

Frana: Are there good cautionary tales in HR software? The prototypical example I always refer to is the Halloween problem that you are probably very familiar with. It's a relational database problem: how do you give everybody a raise of $1,500 dollars who is making less than $20,000 and the payroll register just continually gets updated until everybody is making more than $20,000 dollars. It's a cautionary tale—don't make this mistake. And sometimes they're even more fanciful. You can't pin down who the 'naked programmer' is. But we know he exists somewhere in the Silicon Valley. Are there stories like that that get passed around?

TIME RELATIONAL DATABASES

Durbin: Yes there are—there are lots of those. There are a lot of them that have to do with what's called the time relational database. This is a database that changes over time but yet keeps track of its history so to your example, people get raises from time to time to time but we'd like to know a year ago what everybody got paid and people are being continually hired and terminated. And a lot of the paperwork is late or early. This person is starting in two weeks, that person got fired last week, and the company would like to know at various

points in time who worked for the company. And this used to be one of our things on HR. We'd go in and we'd ask them, the most senior executive we could get our hands on, how many people work for the company? And he'd say da-da-da. And we'd say, 'Are you sure?' And we did a big company—150,000 employees—we converted the database and we reconciled the HR system and the payroll system and they were 8,000 people short. They were paying 8,000 people more than they knew what they were doing. And we sort of went 8,000 people? And they said that's only a 6 percent error. Eight thousand—that's larger than a lot of companies. There are 8,000 people in the company that they didn't know what they did and it didn't really bother them. That's only a 6 percent error, but using our system they could cut that error in half; that would be a big improvement. So a lot of it has to do with this concept of time and how information is morphed over time.

In fact, I'm on the editorial board of the *Harvard Business Journal* and I've been talking to one of the editors and I'd like to find some time to sort of map out the history of the concept of time in databases. Back in the 1960s and 1970s you would have had a flat file where data comes in and you change someone's salary and 'boom'—it's changed and that's it. And then we'd like to know what happened. We'll spin all these things off and do a history file. We still have this concept that time moves on this single plane but we'll spin stuff off to history because we have a time plane. Well, it turns out that that's a very difficult thing since that isn't the way the real world works. So you sort of take a time plane and turn it sideways. Now you say we are going to keep track of the history of things and you tell me what slice of that time you want to look at. That is just a view of this larger concept.

Frana: I wish you would do this because it would be fascinating. It would be a good thing for me to read.

Durbin: It's gone through various evolutions. I'm particularly interested in doing it because I'm supposed to be the inventor of the time relational database and there have been some papers. There is a book written on temporal databases that sort of looks at a whole different thing. And it turns out what you end up doing is looking at various entities as to what their relationship is at the time. Some of them have discreet relationships—you know—it's this way and then it changes and then it's that and then it changes and so forth. Others are much more of a continuum. Some of them have no continuum; they are just dropped into a transaction system that checks on your balance in your checking account. There is no relationship from check to check where you can say there is a change in state. There is an account but that is a different thing all together—it's a different entity. The check entities are just sort of these stacked up fragments that all relate to this thing but don't have a continuity over time. They have a time base because they have an effectiveness but that is a different idea. So there are all these different ideas that have evolved about time and I had a long conversation with Dr. Codd one time about this. I said, 'You know, the problem I really feel you haven't solved is this time problem.' And I went into so many things and he said, 'Those are implementations.' Then I said, 'Well, that may be, but we have a problem because we haven't created a set of standards yet.' The thing that relational databases did was they created a more or less uniform set of standards. Unfortunately, Codd didn't go quite far enough. If he'd gone far enough we wouldn't have all these different implementations, like what's a 'number'? Well a number isn't the same thing in Oracle and Sybase and DB2. They have different characteristics and you can't necessarily just loop stuff back and forth and the function sets are not consistent.

Frana: It would have been hard to think all those things though.

Durbin: No, many of them came up very early on.

Frana: Really?

Durbin: Oh yes. They were in early discussions and you had IBM and Oracle and so forth and the other one who is no longer around anymore. Each one of them wanted to own the standard. See, Codd had the opportunity and the stuff that Codd defined became part of the standard, but then he stopped. And then you had the vendors. Once a vendor had proceeded then they wanted their way of doing it to become the standard because they had their customer base and they had a competitive interest. Once you got those competitive interests you could no longer merge a standard. So the beauty of the relational database is that you actually had somebody like Codd establish a standard for at least the structure of information. It didn't establish a standard for the language. That had to get fought out between SQL and SEQUEL. No, no, that was actually the first language. Another was called SOQUEL. We tried to pronounce it a little differently. It was the one used by the company headquartered in Alameda. There were three splinter groups that came out of System R. The guys that went to Santa Teresa, the guys that went to Alameda and the guys that did Sybase, and then out of Sybase or one of those others came the guys who did Oracle. Oracle was actually later on with QL track.

So each one of them was built. First was the language fight about was it going to be SQL or was it going to be something else. And of course Codd did not engage in the fight but he didn't even get to the point of saying a 'date' is something that is composed of date and time to an adequate level of precision. How do we define what precision we're dealing with? Those things were all abandoned.

Frana: And you said that was because of implementation?

Durbin: He didn't get to it.

Frana: He never really got engaged with System R development either. I understand he really distanced himself from that project.

Durbin: Well that was sort of the dirty stuff. But yet he came back later and said—and I went to one of the seminars where he did this—he said that none of these guys has really implemented a fully relational system. And I felt like saying, 'Yes, because you dropped the ball and didn't pave the way so you can't complain about what road they took.'

CURRENT ACTIVITIES

Frana: Looking forward, where do you see yourself heading?

Durbin: Well, I do consulting. People have been waiting for me to start my next venture. Well, by my count I've already started five.

Frana: And reached your quota?

Durbin: Yes. I really like advising and working with software companies because I go in and work with the management and they go da-da-da-da. And I say, 'Well, I solved that problem in two different ways. Let me tell you about one way that didn't work, and I'll tell you another way that did and what I see about the differences.' And they go, 'Oh that makes sense.' And that's cool. But then when I leave it's up to them to do it or not. So I don't lose much sleep as if I were on the firing line. Actually my current project is I'm writing a novel, a software company novel.

Frana: Really? A fictional account?

Durbin: A novel that takes place in a software company, in a start-up, in an early stage.

Frana: Is this something you have a contract for?

Durbin: No, I'm 30,000 words into an 80,000-word book. I have an outline, chapter-by-chapter, that goes into a great deal of detail and I think it's about 80,000 words.

Frana: Is it a mystery story?

Durbin: Yes, it's a mystery but it has things like the guy who's in to solve the mystery, discovers that there are intellectual property problems. The guy who had supposedly invented the software actually copied most of the software. But then he made so many changes and those changes are probably patentable and patents are an issue. Did you know at one point, prior to about 1980, whenever a software company would go after a patent IBM would file a friend of the court brief on the other side. They fought patents like crazy. Four years later IBM was filing more software patents per day than any software company on the planet. Talk about a change in posture that just snapped the industry.

SOFTWARE PATENTS

But actually now we have a lot of problems in software patentability because of those early times. It was held back for so long and all these positions were created averse to effective software patenting. We now have a much weaker software patent system that is actually quite susceptible to abuse. When I was working on my software patent I was working with a really cool software patent lawyer and one of the reasons he was cool is he used to work for me as a programmer. He had been a lawyer, he worked for me; he didn't like being in private practice. He came to work for me as a programmer. He was one of that group that I said that we did some training. Well, he was one of them and then eventually he went back to law but took the patent bar exam and became a $500 dollar an hour patent lawyer. He was involved in the IBM-Microsoft or the Microsoft-Apple copyright deal. He was one of the attorneys in that one. Anyway, every week or two he would send me some patent and it was either IBM or Microsoft and they were filing these patents that were just unreal. I mean it was like—I think it was published in Knuth. And what you had, particularly at Microsoft was a bunch of kids who graduated from computer science class who had no idea what had been done before and they'd do something they thought was cool and the company would let them file a patent. The patent examiners didn't know anything about software either so these

damn patents would get issued for the most stupid thing. All that garbage is out there. You can't build a piece of software these days without tripping over one of those things. And all it does is give Microsoft the ability to go in and slap you around. Even if you happen to get a real piece of software that you get a patent for, but then you've stepped on all these things of theirs so they get to beat you up. It's a weapon for the big guys and it's really not protection for the small guys.

Frana: Don Knuth is actually quite distressed about that. I was talking to him in November and he says that he has been testifying in Europe because the patent laws are not quite so extreme yet. He was very concerned that some of these low level operations are now getting patents that never deserved to be issued.

Durbin: A patent I remember is one by Microsoft where for debugging purposes they go in and replace an instruction where you put a break point with an invalid instruction and take a program check. I was doing that stuff back in the 1960s. You put in 1401 and IBM designed it so that the halt instruction was a one character instruction so that you could put it anywhere you wanted to and stop the program. That was in 1962. And this was an issued patent.

Frana: So did you have to go back and establish preexisting rights on your software patent?

Durbin: Oh yes.

Frana: You had to do a big search and see what else was out there.

Durbin: We found a whole pile of things. But this is different than that because the software patent for the massively parallel system was a pretty big thing.

Frana: Is there anything else on your mind that you really want to discuss?

Durbin: No, we've actually covered much of the ideas from your subject list. I think we've pretty much covered them.

Frana: 'Just-in-Time' software is something that people are talking a great deal about these days. Is that just another way of getting people to structure programs?

Durbin: You know part of the issue with software is spending an adequate amount of time designing. What is interesting about software is that the better the design, the less code. So I always used to have this criterion when one of my development teams would be working on something—I would remind them of this and then we would evaluate it later. If they were building a new function, I expected that they would end up with less code because what they ought to be doing is finding where the other components of that functionality already existed and consolidating that and abstracting it and thereby reducing the total amount of code so that every release should have more function and less code. This idea of understanding and abstracting and moving things up through layers of abstraction means that eventually you can have very, very powerful components that you ought to be able to go, click, click, click and bolt them together like Lego's and that's the vision of object-

oriented design. How much of that really goes on though? In my experience not a lot because people never really developed those skills.

Frana: The companies that you consult with aren't doing the best job?

Durbin: We did a project at one point—we were in the early stage on this—and we talked to the company and said that we think that the design and the architecture of this thing should be about 50-60 percent of the whole project and they said, 'Oh yes, okay.' We're about 40 percent into the project and they said, 'So how much coding have you done?' 'I haven't done any yet because we are still architecting, designing, and determining how to debug the system before we code it, just like we told you.' They said, 'You're behind schedule.' They didn't get it. So even though you have these practices, when you get down to a real situation, management people don't necessarily manage that way. They decide, we're 20 percent in, we should have 20 percent of the code done. Wrong. So there is a gap between management and the art of programming and we haven't reconciled those two.

Frana: Maybe that is a good place to stop, Gary. Thanks so much. I appreciate it.

Durbin: Yes. This has been fun.