An Interview with

JOHN BRACKETT AND DOUG ROSS

OH 392

Conducted by Mike Mahoney

on

7 May 2004

Needham, Massachusetts

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

John Brackett and Doug Ross Interview

7 May 2004

Oral History 392

Abstract

Doug Ross and John Brackett focus on the background of SofTech and then its entry into the microcomputer software marketplace. They describe their original contact with the University of California at San Diego (UCSD) and licensing the p-System which had been developed there. They discuss the effort required to bring the program to production status and the difficulties in marketing it to the sets of customers. They talk about the transition from 8 bit to 16 bit machines and how that affected their market opportunities. They conclude with a review of the negotiations with IBM and their failure to get p-System to become a primary operating environment. That, and the high performance of Lotus 1-2-3, brought about the demise of the p-System.

[John Brackett requested that the following information from Wikipedia, the free encyclopedia, be provided as an introduction to this oral history interview:

“UCSD p-System or UCSD Pascal System was a portable, highly machine independent operating system based upon UCSD Pascal. The University of California, San Diego Institute for Information Systems developed it in 1978 to provide students with a common operating system that could run on any of the then available microcomputers as well as campus DEC PDP-11 minicomputers. UCSD p-System was one of three operating systems (along with PC-DOS and CP/M-86) that IBM offered for its original IBM PC. Before that, IBM used UCSD p-System as the operating system for its Displaywriter, a microcomputer-based dedicated word processing machine (not to be confused with IBM's DisplayWrite word processing software).

UCSD p-System began around 1977 as the idea of UCSD's Kenneth Bowles, who believed that the number of new computing platforms coming out at the time would make it difficult for new programming languages to gain acceptance. He was particularly interested Pascal as a language to teach programming. UCSD introduced two features that were important improvements on the original Pascal: variable length strings, and "units" of independently compiled code. Niklaus Wirth credits the p-System, and UCSD Pascal in particular, with popularizing Pascal. It was not until the release of Turbo Pascal that UCSD's version started to slip from first place among Pascal users.

UCSD p-System achieved machine independence by defining a virtual machine, called the p-Machine (or pseudo-machine, which many users began to call the "Pascal-machine" like the OS—although UCSD documentation always used "pseudo-machine") with its own instruction set called p-Code (or pseudo-code). Urs Ammann, a student of Niklaus Wirth, originally presented p-Code in his PhD thesis (see Urs Ammann, On Code Generation in a Pascal Compiler, Software—Practice and Experience, Vol. 7, No. 3, 1977, pp. 391–423). This p-Code was optimized for generation by the Pascal programming language, and all the original development was done in UCSD Pascal. Each hardware platform then only needed a p-Code interpreter program written for it to port the entire p-System and all the tools to run on it. Later versions also included additional languages that compiled to the p-Code base.

UCSD p-System shares some concepts with the more current Java platform. Both use a virtual machine to hide operating system and hardware differences, and both use programs written to that virtual machine to provide cross-platform support. Likewise both systems allow the virtual machine to be used either as the complete operating system of the target computer or to run in a "box" under another operating system.”]

Preface

As part of the Software History Center's collection and preservation activities, and in conjunction with its meeting on the history of personal computer software held in Needham, MA, on May 7, 2004, the Software History Center (SHC) arranged for 14 oral histories to be conducted with computer software company founders and other key industry participants. All of these oral history interviews were conducted by historians well qualified by their knowledge and interest in computing history.

The following is a list of the people who were interviewed together with the name of their interviewer:

John Brackett and Doug Ross, interviewed by Michael Mahoney
Dan Bricklin and Bob Frankston, interviewed by Martin Campbell-Kelly
Dan Bricklin and Bob Frankston, interviewed by Paul Ceruzzi
Jerry Dreyer, interviewed by Thomas Haigh
Ben Dyer, interviewed by Nathan Ensmenger
Dan Fylstra, interviewed by Thomas Haigh
Gary Harpst, interviewed by Tim Bergin
John Imlay, interviewed by Bill Aspray
Luanne Johnson, interviewed by Janet Abbate
John Landry, interviewed by David Grier
Mike Maples, interviewed by Nathan Ensmenger
Seymour Rubinstein, interviewed by Jeffrey Yost
Jonathan Sachs, interviewed by Martin Campbell-Kelly
Oscar Schachter, interviewed by Thomas Haigh

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

On January 1, 2005 the Software History Center merged with the Computer History Museum, and its work is continuing as the Software Business History Committee as part of the Museum's activities (see www.softwarehistory.org).

Software History Center Oral History Program

John Brackett and Doug Ross Interview

Mike Mahoney: This is Mike Mahoney and I am interviewing John Brackett and Doug Ross on the subject of SofTech's Microsystems' p-System and Pascal. It is May 7, 2004 and the interview is taking place at the Sheraton Needham Hotel in Needham, Massachusetts as part of the Software History Center's Oral History Project. Since SofTech was an established company, I'd like to start by talking about the business that SofTech was in and then the decision to enter the microcomputer market, in particular the choice of p-System, which I take it was the main product of that new division.

Maybe we could start with you Doug - you were one of the founders of SofTech. Tell us about the business that SofTech was in and then we'll move to the decision to go into the microcomputer market.

SOFTECH BACKGROUND

Doug Ross: I'll try to stick to the versions that have to do with actually setting up and responding to this opportunity for doing things for the microcomputer industry, because we actually went back well before that.

Mahoney: You could start by characterizing what SofTech's established market was as far as products, and with whom were you working.

Ross: SofTech is known by its trademark: The Software Technology Company. That's what we'd mainly grown on and had already under our belt. Both the APT language and American Control for which I'm noted -- they were essentially so primitive a language that it didn't really get us anywhere in terms of solving the problems of the industry that needed it. It was technologically not a major feature. The language that made us most competent was AED -- Art of Engineering Design, which we did to address the problems of making computer aided design systems and computer aided design projects.

We also then combined that -- that was the first software engineering language with complex data structures and modular program designing-- integrated packages that could be used as building blocks to make a complete translation system. We also did this in a totally machine-independent manner as far as we could, right from the beginning. This was because in a precursor language, we had started with what was called the Bootstrap language and it ran on both the TX0 computer, which was the tiniest and the first transistorized computer -- it came from Lincoln Laboratory. And in parallel we developed this Bootstrapping language on the IBM 709. So that started us clearly on this

mode of being interested in machine independence because of course there were only the large mainframes in those days. There was not even a mini computer on the market, much less a microcomputer.

So by the time we came to completing our computer aided design work at MIT, we had produced this AED compiler and the system for doing it. We called it software engineering language and it really ran simultaneously without major change of its source language on all the major big computers. So we undertook to have a company doing this, which we called SofTech.

Mahoney: Basically I'm interested into what you were then were moving into -- this new area of microcomputer software and what experiences you had built up as a company - how they steered you to a particular market or made you feel that that was the way to go...

Ross: Let me try to show the origins of that, because you see, John Brackett here, for example, was at MIT at the same time that we were and was a user of our AED system, to the point where we both had portions of projects running on Project MAC.

John Brackett: SofTech was about 9 years in business when we made the decision regarding microcomputers. Through that time the Company had done three things. One was that it sold improved versions of the AED software that was developed at MIT. We increasingly moved into using it to build things for people, largely because these were very sophisticated software engineering tools and many companies liked what we had done before. However, training their people was maybe subsidiary to getting some job done. One area where the company built a good reputation was in building compilers. These original tools were compiled building tools. We built the Jovial compiler that was used in the first airborne computers on the B-1. We got the contract to implement the first Ada compiler.

So we did a variety of things in system building tools, based on this technology. The second area came as we grew -- a lot of people wanted to be trained in software engineering and we were fairly significant at that. We did a variety of commercial projects; we built things for people. So you could think of us as both a technology company in terms of tools and building tools, and using them to build things for people.

During the 1970's our business evolved increasingly into the defense community. There was a lot of discussion in the company of whether or not we wanted to be as much into the defense business as we were, so we started to work in the late 1970's on what I would call product opportunities that weren't dependent upon the amount of government spending.

The second thing that was happening was that it was easy to see that in products like airborne systems and instrumentation systems, people were starting to use microprocessors. So our tool building people got interested, for example, in the 8086, basically for embedded systems, because they were the first mass-produced chips that could replace the custom chips in some of these

airborne systems. In other types of systems, our customers were using our tools. So, over time, we built the tool building business, since customers wanted our tools, including people who would be using small microcomputers. So over time, we got smart on some of these microcomputers. Not personal computers, but with Embedded systems.

I think the last thing I'll say is that there was one guy in the company, named Al Irvine, who was one of the first of what I'm going to call the "PC hobbyist" types, who went and saw these early personal computers -- 1977 vintage. He had an Apple I board-level computer, the first product of Apple. So he was the first one to show any of us what events could happen. And that's my view.

SOFTECH MICROSYSTEMS

Ross: Now I can take the ball back. Thank you very much, John. But a key factor was that SofTech always had a strong growth path. We were certainly compatible with the same direction that John was talking about. It included having not only customers all over in many places, but also opening offices in new locations. So Al Irvine, for example, had joined us originally when he had worked at NCR in San Diego. He was in charge of the Century Operating System. But he learned about our technology so he moved himself from California, where he had always been, and joined us in Waltham. He went back out to our San Diego office which was then an opportune locale as well for when the Regents of the University of California needed to find a company to license UCSD Pascal.

Mahoney: Was there a specific decision to pursue this market?

Brackett: I think we have slightly different views on this. I was very interested initially in our type of development tools for these embedded processors. Al was much more interested in what he thought would be the mass of these small computers growing up. So Al was the salesman for the idea that the UCSD p-System and Pascal were for people who would write the applications to propel these small computers forward.

Mahoney: So he brought p-System to your attention?

Brackett: Yes, he brought it to our attention.

Ross: He sure did.

Mahoney: And how did it come about?

Brackett: He was in San Diego, and a lot of people at the University of California at San Diego were around. The office there -- I guess we had a dozen people doing custom software -- knew some of the people who were working at the University. Many of them were people looking for jobs when they got out, because they were undergraduates or master's candidates.

So Al would come in contact with some of the people who eventually were hired to help maintain and enhance the p-System. So I think we can argue that we went out looking for the best business opportunity in personal computers.

Mahoney: But you had already decided that that was the direction in which to go?

Brackett: I felt we needed to support our customers who were building embedded systems. That's when I started looking at tool opportunities for these small chips. So we knew the underlying hardware technology but we hadn't - up to the time Al brought this opportunity to us -- thought about getting into the personal computer business.

Mahoney: So that decision was made. Was that the point at which Microsystems was established?

Brackett: Microsystems was a wholly owned subsidiary established in 1979.

Ross: John became president of SofTech in –was it 1975?

Brackett: 1976.

Ross: And did a great job with this early expansion of the company in the sense that I was describing it. These embedded systems came from having worked with the Navy as well as the Air Force. We did work with tanks, too.

Mahoney: And Al Irvine's position?

Brackett: Al was the manager of our California office. They had contracts for custom software with people like the Dept. of Energy, Lockheed -- in SofTech's traditional business, basically.

Mahoney: So he brought the p-System to your attention.

Brackett: Right.

Mahoney: And you found it attractive?

Brackett: Let me put it this way. It was an alluring concept that you could have reasonable quality development tools that ran on the smaller computers. A lot of people were interested in it because, very frankly, there was almost nothing for building real applications comparable on personal computers at the time. The time was late 1978, early 1979. But I would say to a large extent, the fact that the University wanted a licensee and we didn't have to develop it, was an attractive part of it. It wasn't as though you were going to define a business concept and then build a product from scratch. Although it was university quality software it was largely developed to a point

that it was stable.

Ross: I was Chairman of SofTech and John was President. So I was very intrigued by what was being brought up here. So I went out and spent a week out there in a hotel and got totally absorbed in the system -- to the point where the hotel people were saying: "Won't you try the swimming pool?"

I was just so smitten by it - that we just had to do this - and we didn't plan business that way -- the way that most do perhaps. Because we built things in stages from the inside out, but always were heavily targeted toward what the actual user needs are. And I was just intrigued by everything there.

PRODUCTIZING THE P-SYSTEM

Mahoney: From what I understand about that system and p-code, essentially you build your program and your program is translated into p-code, which is then executed using an emulator.

Ross: It's called p-machine.

Mahoney: p-machine. And I listened to you describe the virtues of AED. Did you recognize a kindred system, as it were?

Ross: Very much so. Yes. I didn't know the people -- I didn't know Ken Bowles, who was doing it, till I got out there, but I was intrigued, and Mark Overgaard, who was the primary programmer for building the system, and many cuts above the average guy you'd find just mucking around in a college operation. It was first class.

Mahoney: So you recognized the product, as it were?

Ross: Yes, we just adopted it. In fact, later, I did buy the Terak 8510/A computer that Ken Bowles developed his side of the system on -- it had the best on-screen graphics because it had a memory that you could write and then display from.

Brackett: We should mention -- most people won't recognize – that it was a PDP-11 board based computer. It was re-packaged by this company, Terak. So it was a 16 bit computer and it had nice graphics. It was very expensive.

Ross: It was a real work horse for the field, and later I bought that actual machine from Ken Bowles and it now resides at the Computer History Museum. He carried it all around -- it wasn't really portable, but he had it in 3 big boxes that you could get on the airplane and that's the way it worked.

Mahoney: Two questions immediately pose themselves and they're related. I want to pick up,

John, on your remark about you bought warts and all and ask you if you can identify the warts. What did you think had to be done with the product in order to make it suitable for the clientele you had in mind?

Brackett: I think there were three things we had to do. We knew it right up front. One was this product was mostly developed by students and clearly needed to have added all the things like comprehensive suites of tests and configuration management so you could enhance it, re-test it.. There were two or three versions that were slightly different. The version you got for one machine which they made last year, would be different from the version for this year. So configuration management and regression testing were clearly a problem.

The documentation was like most university software - it got written in bits and pieces by different people, and it certainly wasn't something that was the quality that any credible software products company would ship.

The third thing was -- and I don't believe we ever dealt totally successfully with this -- that there were design assumptions in the p machine that should have earlier been looked at closely in order to try to deal with some of its efficiency issues. We had enough to do to what I call 'get the quality product out there', and document it. We didn't deal with some of the efficiency problems early enough, I think, for one simple reason: Until 16 bit machines came along in the commercial world, it worked better than any of the software on these 8 bit machines which didn't have much memory. So the problem was largely ignored up until maybe 1981 or 1982 when 16 bit machines became much more common.

Ross: I think to leap ahead to the tail end of this story -- Apple, all the way through, had acquired a parallel license. We thought, at the corporate headquarters at least, that we were getting the sole license, including Apple, but that turned out not to be the case. And that did give us a hiccup.

Mahoney: So you developed the product at the same time Apple was.

Brackett: There was a code base which had some things in slightly different versions, etc. from which Apple got a base version and we got a base version. Apple did its own development and phased it to their machines and added a few features that didn't exactly appear in the same way in the version that we developed over time.

Ross: Let me just put in small point. We got to the point where Apple really wanted to have the whole system on the Macintosh. Their branch of the development and fixing and so forth versus ours led them to the situation where they were only able to compile on their larger, never successful Lisa computer. Apple came back to us and did pay for, I believe, a version of our system that would run on the Macintosh. It was the only thing that could do that, and I think that says something about our technology versus theirs.

At the end of the story, where IBM was going with Microsoft in the business of supplying multiple manufacturers with multiple products, we disappeared overnight, actually, in a very quick time.

Mahoney: So basically it ran on the 68000 as a main system?

Ross: Yes. We did it in just four or five months.

Mahoney: So let me recap. First, if I have your three points, one was to impose proper configuration management.

Brackett: And testing and get to the point where the software was of "industrial strength."

Mahoney: Including documentation, right?

Brackett: Right.

Mahoney: And then you said there were problems in the design assumptions.

Brackett: Yes, one of the problems was that we inherited a fully designed product from the university. The big question always was -- there were a significant number of people out there using it, so would it stand evolving in a compatible versus an incompatible direction? We inherited a fully designed product with the design assumptions of the original designers.

MARKETING THE P-SYSTEM

Mahoney: Let me again suggest a branch here. Did this become the tool you were looking for, for your embedded customers?

Brackett: No. Because the embedded customers, except in very few cases, couldn't live with an intermediate code version, largely because most embedded companies only sold one processor. Some of them, like Lockheed who had airborne processors, were going to have one line of custom hardware they built with the same chip in it, so that they would provide support for the Air Force for lots of years. So they had to generate efficient code for those processors. So p-Code really didn't have much of a home in the embedded system market.

Mahoney: How much of your established customer base could you offer this product to, and to what extent did you have to create a new customer base?

Ross: The customer base here was for small individual business owners, used as a personal computer. So even though the original attraction for technologists like Dr. Brackett here -- getting those things right was important. It actually was responding to quite a different set of people. I think we did as much growing in how to market that product and how to put on good shows at Comdex year after year and how to set up a user group that would really build on our previous

experience with user groups from mainframe area. We weren't complete greenhorns, but on the other hand, it's quite a different flavor to go out and talk to people who are carrying inventory in their stores.

Mahoney: What did you already know and what did you have to learn?

Brackett: There were three different markets we were selling to. There were the people who wanted to buy UCSD Pascal, and we also had our FORTRAN in a box to use on their computer at their company. Because this was a good application development environment, we were also selling it to application developers who wanted to be able to distribute our operating system along with their application on a diversity of machines because of the architectural diversity of the late 1970's, early 1980's.

And the third one, the biggest challenge, were the computer manufacturers who saw there were applications out there and who wanted to offer the p-System as one of their operating systems, so that they could attract application developers to support their machine. And that included people like Philips in Europe, Texas Instruments, Commodore, Osborne. We'll come back to IBM.

Ross: And Radio Shack.

Brackett: So we had three different customer bases - the big OEM's, application developers and the end users. It turned out that this had a big impact on what we had to work on. In the beginning, we knew much more about how to work with the big OEM's, because they were the type of people that SofTech had basically done business with.

So dealing with what I'm going to call the individual application developers in a garage or a small company, was definitely a new business. And we had never sold anything that retailed before, so selling to people at a computer store was a new business totally for SofTech.

Mahoney: And you didn't have experience in that.

Brackett: No.

Mahoney: So what did you do?

Brackett: The problem we fundamentally had, which I think we didn't recognize for while, was the skills required to sell to the OEM's - Philips, IBM, etc is fundamentally different from selling to the retail channel. We were never as successful, in my opinion, in building the capability to sell to the retail channel. Most of our business in terms of dollars of revenues - it was in selling to the big OEM's and helping the application developers enough so that they would produce the applications that the OEM's would want to have on their machines.

There was always the retail business, but it was always over there in a corner and I don't think it ever

accounted for more than 15 or 20% of SofTech Microsystems revenues. This was largely because Apple had the biggest selling machine with Apple Pascal in their own box. People like Radio Shack were not that interested in putting program development tools on their shelves. They were much more interested in those days in putting simple word processors and games on their shelves. So there wasn't anybody else selling at the mass market level as Apple did that particularly understood what something like Pascal could do for their machine, independent of applications from business software developers.

So the retail part was a part that definitely never did very well. That's a good summary of that.

Mahoney: How about the application developers? How did you reach out to them?

Brackett: Mostly on an individual basis and many of the bigger ones had their own marketing interface inside the company. Software Publishing was #5 or #6 in software sales and was the biggest of the p-System application developers, and they had grown up on things like Apple on the Z80 machines, but they wanted to have one version of applications that ran on all of these machines.

So p-System was a logical choice for them. They were very good at marketing to the retail channel -- probably one of the best at the time. Fred Gibbons, the president, had come from Hewlett Packard but he brought with him some people that had sold things into the retail channel. So Fred was doing the retail channel well, so he became the best of the p-System developers.

And there were also companies like Context Management Systems that built perhaps the first integrated product that involved a spreadsheet, word processor and a file system. So there were p-System application developers, and we had people who could give them help to effectively use the p-System and Pascal as an application development tool.

Mahoney: Was p-System included in their application or just used for development.

Brackett: You have to be very careful. The p-System, the operating system, typically was included by them on their disk for a particular machine. In some cases, if the manufacturer like Philips, put it on every machine, application developers didn't have to do that. But in most cases, since they were going to sell the applications retail, they put it on the disk along with the p-codes for their application.

So if you want to think about it, the interpreter lived on top of whatever the native operating system was and then they packaged the p-code. And one of the big attractions was that until hard disk came, you could put a lot bigger program on a floppy disk in p-code than the native code that came out of an assembler.

It was very compressed, because the p-code instructions were at a much higher level than the instructions that the 8 bit machine used. So the appeal to the application developer was that you

could ship your product on a disk and give everything. The user basically installed it, loaded it, and everything you needed to run it was there.

TRANSITIONING FROM 8 BIT TO 16 BIT MATHINES

Mahoney: But of course it got passed by, by the 16 bit.

Brackett: We should talk about the transition from 8-bit to 16 bit machines. It's important to understand the application developers that we recruited in the period of 1979 to 1982. If you wanted to sell on a variety of machines without a lot of what I call assembly code for each machine, we were, in some ways, the only game in town.

Mahoney: So they came to you.

Brackett: Pretty much.

Mahoney: What about the transition to 16 bit?

Brackett: I'll say two things before we get into details. One of the issues that surfaced very early on -- if you think about the p-System, it was an operating system for an 8 bit computer effectively, because it assumed 8 bit addressing for memory and a variety of other things.

There were some people who believed we should have developed a 16 bit interpreter that also could run the byte codes for the 8 bit interpreter. And that it could address bigger address spaces.

And the second one was that in a 16 bit world where the machine could address a very large amount of memory, that interpreters were not the way to go because of the execution speed penalty. Now, to make a very long story short, there was discussion of these issues within Microsystems and with some of the people at the University. Very frankly, no one ever made a decision to do it, partly because it was a tremendous amount of work supporting the application developers and the OEM's who wanted to put the p-System on their 8 bit machines.

It was basically 1983 before a majority of machines were being shipped with 16 bit processors. So the problem basically was -- and this is a very simplistic view – that by the time it became clear there was a need, the need had been solved by applications going direct to the hardware. We might well have postponed the end of the p-Systems' technological life, but it would have died because the hardware had such large amounts of memory, and once people started to exploit that - it's not easy to do that in an efficient way with an interpreter.

Mahoney: Let me back up for a second, because you said you were being kept busy by your applications developers. What were your relations with your customers? What kinds of requests were they making and what kind of enhancements and changes and so on were you asked to make?

Brackett: Firstly the applications developers, as they tried to build bigger applications, had two problems, one of which was how much you put on the disk, because hard disks didn't really show up until 1983. So therefore what they were looking for was a way to do the following: get more memory room for the application and to have a way where you could first put all the application pieces in a library on disk, then have the application bring in the pieces as needed.

But eventually there was an approach to independently compile procedures and create a big application that loaded application pieces as needed. Many of those things made the p-System an increasingly powerful development environment. And products like Context MBA, the biggest program ever built with the p-System, today would probably be like 50,000 lines of C. They could ship on two 360K floppy disks. So applications developers, as they developed more powerful applications and built bigger things; needed better support for separate compilation, dynamic loading. A whole variety of things like that kept the developers busy because we all realized that if people stopped building applications for the p-System, the OEM's that were licensing it would have a lot less reason to license it. So we were driven in a lot of ways by the needs of application developers.

Mahoney: But your own developers who were doing this, were these people building what they knew from the old business or did you have to go out and find different developers who had different kinds of knowledge?

Brackett: Yes to both questions. The people who put the p-System together we recruited from UCSD and some of them were already staff members, some had been students that were graduating so we had hired them. However processes like configuration management and testing and documentation, were widely supported by people from the California office that had been part of the old company.

So in some ways, like all the processes, the old Company would support Microsystems.

Mahoney: And that worked smoothly?

SHIPPING AND ENHANCING P-SYSTEM

Brackett: Pretty well. I think the quality of what we were shipping by about 1981 was up to the quality standards of what the parent company shipped - which was really good for the time. We had a PDP 11/45 timesharing system on which we kept all the files of our configuration management system. It was a state of the art software development environment that was competitive with the type of things that the main company would have used for similar type projects.

Mahoney: So was it a SofTech product?

Brackett: Well, we never sold it as a product, but it was definitely intended to simplify the development and maintenance of the p-System.

Ross: Our style of technology.

Mahoney: Did you make any major enhancements to the p-System? If I hear you correctly, the emphasis has been on making the system more efficient, more compact. Were there any enhancements beyond that? Was the system able to do things that it hadn't been able to do before?

Brackett: Subtle things, for developers primarily. My view basically is that people built much more sophisticated application through p-System, but the underlying system didn't change that much other than these application developer support features.

Ross: Yes, we did have a discussion - I don't remember how far we got toward it- to address the efficiency problem because if you analyze a well organized application, whatever the area, you'll find that most of your time goes into just a few very tight loops. So if you do a tune up of where are the boundaries between the component parts of whatever your application is, then you can pick those out and do those parts, not with the interpretive P machine, but directly on the machine and the interface routine could call it in and out. But I don't remember adding the documentation for the re-user, but maybe we did.

Brackett: If you do it right, assembly language procedures called by the interpreter can be used. So some application developers insisted on that; but it seemed much simpler concept than it turned out to be. You had to make sure that the execution of these programs didn't have unexpected side effects.

Ross: Very strict coding at the machine level so that you don't have this make-believe p-System action -- all of a sudden you have a quirk.

Brackett: That became important for what I call high end developers who were trying to build bigger and bigger applications on either the 16 bit machines when they first came out or 8 bit machines. The people who bought it to learn Pascal, universities to train people programming, never got into a lot of those facilities, but they were essential to the growth of application development groups.

Ross: It was in our original AED compiler, too; it was called Mashee and it really could mash things!

Mahoney: Did you play an educational role for your customers?

Brackett: We did some on site training - basically for the big OEM's - about what was under the hood and how to reduce the size of applications and to increase their performance. We talked about how we would continually support the application developers by adding more and more things that made it easier to build bigger and bigger applications.

Ross: I'd like to mention one more thing, which came out of the product development within subset Microsystems that was underway. Just before we had to shut everything down -- because the marketplace changed and we didn't have our OEM customers and so forth -- we were about to demonstrate it. I forget whether we actually did show it in the last Comdex, but Al and Mark Overgaard and probably other people as well- had been working out the latest thing in networking. Only the P machine could directly address, with one set of source code, all the different kinds of things that could be on a network out there. The name of the new product was going to be Liaison, and what was magic about it was that you could plug it in and take out notes on the fly. You could also -- when you needed to have some services to go along with a computing note, something like a particularly large database or whatever -- you could do that because it was all one p-System spread out over the whole place.

I don't remember the details of the other features that enriched the whole concept. I do know that these features that were then ready for delivery in the next year and a half -- if we'd been able to continue- didn't start to show up in that level of sophistication of integrated design -- I haven't really seen. You can't see it through Discover Web.

Mahoney: What about developing applications yourselves? You'd been in the business of developing applications originally. Was there ever a thought about developing p-System applications yourself within the company?

INVESTMENT IN APPLICATIONS

Brackett: We were much more inclined to consider acquiring some of the small application developers. Although we had good product ideas, for the simple reason that the time to market was so important, it was much easier to look at providing things like configuration management, testing, and making industrial quality software from an application that somebody else had developed.

So there was a lot of discussion about going into what I would call a major investment mode for acquiring, repackaging applications and selling them, both through our OEM's and the retail channel.

Mahoney: So what you essentially would do for some of these application companies is what you had done for the p-System, which is bring them in and show them how to do it.

Brackett: Right. And possibly acquire the company. Some of these people didn't have much cash; they saw SofTech as a company that would take a big company approach to making their idea successful. Even though SofTech was only maybe a $30 million a year company, that was big to them.

So there was discussions about should we get into that business. There were pros and cons. First, if we do this with some application developers, we're driving away the other ones. I think Microsoft has proven that if you do enough application support to developers, you can compete against them.

And most of the developers will decide to do something other than compete directly with you. But we might drive away one of our key constituents by being in that business. So that was one thing against it.

The other one was - it was cash intensive. You were basically going to pick up these applications and then you were going to have to package them, distribute and market them and so on. And this is one area where the business was fundamentally different from SofTech's main business. Most of SofTech's main business was doing things for the government and big companies. Send them an invoice every month, they pay within 20 or 30 days and little cash investment is required. In the application product business, you're talking about significant investments that may or may not be paid back for quite a period of time.

It was becoming clear that this was going to eat up -- if it was done -- a large fraction of SofTech's available cash. And SofTech was going to have to raise more money or recognize that it would have to live on a lot less cash after it invested in these applications.

The pro was that it was becoming clear that people were buying computers because of what it could do for them whether it was inventory control, whether it was spreadsheet, word processing, whatever - that's why people were buying these computers. And system software was necessary, but it didn't motivate most people to buy, other than probably the universities.

I was strongly in favor of getting the application business. However, basically at SofTech Microsystems, we had no way to finance it.

Mahoney: No cash pool of your own?

Brackett: No one will finance that. A wholly owned subsidiary couldn't go out and get venture capital outside. So eventually the SofTech Board of Directors passed on a large cash investment into the applications business.

Ross: We did, at the parent company, do such things. I'd like to mention two examples. One is for General Motors, starting very early in the early 1970s because I had worked closely with them in our AED project. I was just introducing SofTech and they said just as I was leaving,:" Maybe you can help us with something." Then they described this ongoing project they had had for 6 or 8 years, with people drawn from all over the world in their General Motors high up research work.

They were looking for what turned out to be a spreadsheet which they ended up developing with our help. It would not have been possible without us working with the parent company this way, but by working with them as a team, you see, that approach worked on the big level. But the product ended up doing all the things with these very massive files and so forth. They literally used to buy Multics computers instead of Apple computers for that one, and we had many years of working with them, maintaining that big system.

The other situation that took a similar bent goes back to the MIT roots, and the computer and design project. One of the applications it was used for was building a CTSS system called CIRCAL, Cersertic Calculation. And that was a joint venture between Philips and Raytheon and ITT. So after we started SofTech, Philips came and said to us, "We'd like to get a stronger calculating circuit analysis calculation package in there, and go on and build this thing up. Would you be interested in doing a joint product development with us on that?"

We said," Is Raytheon interested, too? Well, let's go check." Well, they were on and off, on and off. So we ended up doing it just with Philips to begin with. But we did get the world's best circuit analysis, which still is used to this very day. It's called SPICE, but was built into our AEDCAP product and developed in conjunction with our customer.

You see, they had the marketplace; they had the need. So why we didn't do this approach in the Microsystems form is because we were really brainwashed into effectively having equal partners in the endeavor and having them take all the risk because they were the ones that actually had the product need.

Brackett: So the fundamental problem is the business model for a product company, and resistance in terms of cash and investment. It was fundamentally different from the software development business of the parent company, and a wholly owned subsidiary inside a parent company with a much different business model has problems. These were problems that we didn't fully realize early on. But, as it became clear the money in microcomputers was going to be in applications, that became the big decision for the strategic direction of Microsystems.

Mahoney: When was this decision made?

Brackett: Effectively, the biggest acquisition opportunity was passed over in late 1981 or early 1982.

RELATIONS WITH IBM

Mahoney: What was the interest of IBM in the p-System?

Brackett: Let me start out with a little history. I went out with a couple of our people nine to twelve months before the IBM PC was announced and we met with Don Estridge, who was the manager in charge of the Entry Systems Business, which was responsible for the development of small microprocessor-based systems for 'tiny' business and personal use. The computer was to be built using "off the shelf" components and the design would be publicly disclosed, something the company that had never done that before. Estridge, by far, was the business brain child, but he had legal assistance in software procurement. Legal insisted that you would ship with that machine only the software that IBM had a copyright to. That was IBM corporate policy. We took the position that we would be glad to give them royalty reproduction rights to our things, but we couldn't give

them an unlimited copyright because the Regents owned the copyrights. So we could not give them something that IBM Legal could say that IBM totally owned.

So we had a copyright issue from ground zero that we could not satisfy IBM Legal. In addition, they perceived the market for educational sales of the IBM PC as being questionable. They had no idea that they'd compete against Apple in the education market. So they decided they would like to be able to sell a UCSD p-System and Pascal bundled together with their machine. But they didn't think that was an operating system. They thought of it as Pascal with whatever you had to put on the disk in order to deliver it. So they priced it as a language processor package. Not that much different than Apple priced it. All of that was because they thought of it as a language processor package.

They never perceived the UCSD p-System as a possible operating system. Because of their time to market, there was no possibility to accept that idea of putting the interpreter into hardware. You could have had a support processor that executed p-code but they were trying to build it in an amazingly short period of time. They wanted a native operating system and Gates was willing to do it for them.

Mahoney: Because I had one of those PCs. I was always amazed it didn't have a hard disk in the basic machine.

Brackett: There was the concept of associated chips that ran the p-System well. It turns out at that time Western Digital was already building a chip called the MicroEngine that ran p-code.

Ross: Western Digital made all the DEC chips for the LSI 11 that ran in the Terak.

Brackett: So it was technically possible, but not on their schedule and not with the copyright issues involved at the University. So we were Pascal on their machine; there was Microsoft Basic and those were the languages you could get along with an assembler when they shipped the machines. Maybe there was FORTRAN, but I don't think so until later.

So the p-System was never regarded by IBM as an operating system that they could sell under the terms that Legal had forced on this project. What happened basically was once this machine was out, the application developers who decided they wanted to support it, had not much choice but to build their applications on top of PC DOS, because PC DOS came with the machine.

So if you wanted to sell applications for the IBM PC you could have one of three choices. Go on top of PC DOS; force your customers to buy the $260 package, or put the p-System on your disk of all your applications. The people that supported p-System applications in the first year or so took the third route. For the interpreter they either paid us small royalties or got some sort of a limited license for a fixed fee.

THE DEMISE OF THE P-SYSTEM

The difficulty was that -- let me try to say this in the best possible way -- for the first year, applications developers didn't complain too much about the inefficiency of interpretive execution of their application. Once Lotus 1-2-3 came out, it established a performance level for business applications that nobody other than programmers writing in native assembly language for the machine could compete with. And people at Context Management System basically had their applications blown away because in the time you could do a certain job with Lotus 1-2-3; they were still loading. So in my perspective, Lotus really changed the expectations of computer users to a level of speed that 8 bit machines couldn't do -- they wiped out everybody with 8 bit machines except Apple.

Application developers could move fairly quickly over to the PC DOS. People like our biggest application developer in terms of revenue, Software Publishing, moved very quickly to dump the underlying part of their application in the p-System and recoded it basically to PC DOS interfaces. So people who had applications just changed the underlying structure and kept on selling. But notice, the reason that most of our OEM customers like Philips, Commodore, Texas Instruments and Osborne had the p-System was to sell applications.

Mahoney: If there had been enough time to make p-System the operating system for the IBM PC, would Lotus have had the same effect?

Brackett: Well, applications would have been much faster with a MicroEngine running the p-code as its instruction set. You could get reasonable performance from the chip which I don't think was seriously engineered for performance. It's hard to speculate on how fast, but certainly a lot faster. Also, IBM would have put its name behind the p-System, which would have pumped up application developers to be there too. That was the drum Bill Gates followed -- all the IBM competitors that came out, such as Compaq, had the ability to run applications that were built for IBM PC.

Mahoney: So it became clear that p-System had run its course.

Brackett: I'll let Doug answer that, because I left the company in mid 1982.

Ross: Yeah, but I'm glad to have John so involved in talking about this because although I was the Chairman of the Microsystems board, as well as SofTech when we first acquired the deal with Regents, John was President at that time. He moved out to California to see through this major undertaking for the company. And at the same time, Justice who had been with SofTech for some time became the President as well as the Chief Financial Officer. So at that point, I didn't have the excuse or capacity to go back and forth and visit San Diego as often as I had been, but on the other hand, I did acquire the Terra Atkin. I had my own research penthouse in a separate building, so I was taking off on an application type pursuit. So I can't really fill in the chronology except for just a few pieces.

The way I understood the situation at the time, it was very clear to the whole body of OEM customers which were the main source of revenue for us, that with IBM going off and just being a part of the territory, they had to acknowledge that. So I just talked about it as Bill Gates having IBM in his hip pocket. And it turned out that was the way to look at it.

Mahoney: So basically the OEM people and the applications people followed Big Blue?

Brackett: I would guess if you had to put a date, it's when Lotus shipped in early 1983. Because Lotus shipped a volume of product in the first six months that no PC application had previously shipped. During 1982, when there wasn't really a good hard disk system for the IBM PC, performance was driven primarily by floppy disks. Once IBM came out with their hard disk machine, plus Lotus shipping 1-2-3, even if it ran with floppies, it became clear that that was the winning computer.

So I would say the first half of 1983 was the time when application developers decided they had to follow IBM. And by the end of 1983, and being outside the company then, it became clear that there were going to be two standards: Macintosh and IBM PC and the 8 bit computer manufacturers, the smaller ones, were going bankrupt by the month.

Ross: John and I were still connected though, because he invested in a company that had a board with an 8088 CPU, memory, and graphics for a Apple II computer in order to be able to run programs intended for the IBM PC.

Brackett: We basically had the add-on kit that you could use to run IBM PC applications.

Ross: And your Apple.

Mahoney: Did you use p-System to develop it?

Brackett: No, it was MS DOS on the board. There was a large Apple II base out there, and Apple was still aggressively selling the Apple II. There were two fundamental strategic problems, one of which was that PC application companies that were selling an IBM formatted disk did not want a second SKU in the retail channel for our system. IBM applications did run out of the box, if you already had it in a format that could be read on the Apple II.

And the second thing was that -- and this only became known later -- that within Apple, Steve Jobs, who was running the Macintosh product, was actively trying to kill the Apple II. And our expectations were that the Apple II was going to continue to sell for a fair period of time, given that Macintosh was more expensive. But Jobs made sure that we did not get an OEM deal with Apple, because it would have extended the life of PC application software on the Apple II. So it was a bad investment.

Mahoney: So what happened to p-System?

Ross: The p-System and Pascal business was sold to Pecan Software of Brooklyn. They maintained the operation for several years. I haven't made a recent check whether they're still in that line of work. I don't think so.

Brackett: I think they sold primarily to universities and individual hobbyists and such in the retail channel.

Ross: I think so. Yes.

THE P-SYSTEM LEGACY

Mahoney: What is the p-System legacy?

Brackett: I believe the legacy includes the idea of using a virtual machine in Java, because it solves fundamentally the same problem: the ability to run applications on diverse operating environments as well as CPU environments– such as Solaris, Windows, Macintosh, etc, with chips that are 1,000 or more times faster. The overall performance cost of interpretation in Java is at a much more acceptable level. So I think the p-System influenced the design of future technology.

Mahoney: I've asked all of my questions. What did I forget to ask?

Brackett: I'll just throw out one more comment. The p-System was ahead of its time as a development environment. It had full support for things like independent compilation, putting together large applications. That was a good sign. The problem was nobody in the late '70's thought about building really big applications for personal computers. The problem is it wasn't really scalable to the size applications people were trying to build by 1983, 1984 -- to say nothing about the size today.

So in some sense it as well ahead of its time for application developers from the development point of view, but the design assumptions were not intended to scale it up to say 100,000 lines of code running on a PC, vintage 1983 or 1984.

Mahoney: Is this what you referred to as the design assumption of the virtual machine?

Brackett: Yes.

Mahoney: Do you want to say more about it?

Brackett: First and maybe most important were the issues about how it addressed data and how much data you could have. There were probably ways to solve that for the 16 bit world. People argue about that.

The second thing was that there was nothing built in at the beginning for the idea that you could have an interpreter within which certain operations were done in assembly. In other words, instead of interpreting certain things, the equivalent of Java "just in time" compilation could have been put into the p-System. The difficulty basically was people were not thinking, vintage 1978, about building 100,000 lines of high level code to run on personal computers. Gates, I will say this -- Gates was never worried about building big applications because he believed the hardware would catch up. Microsoft Windows, which was first shown in 1983 and Dan Fylstra's VisiOn, both were far ahead of the hardware. VisiOn didn't have money to wait until the hardware caught up. Bill Gates says he spent several hundred million dollars before Windows really exploded in about 1991 when adequate hardware was available. So if you think about this -- 8 years in there.

SofTech did not have the money to stay with this until the hardware caught up with the p-System. Microsoft is the only company that could finance several hundred million dollars of waiting for the hardware to catch up with a technology, given the tremendous cash flow from operating systems and applications.

So even if they had designed it with different assumptions in 1978, certainly SofTech did not have the money to wait for the hardware to catch up with the technology. So in some ways, just like VisiOn, the p-System was far ahead of the ability of the underlying hardware to support it. It wasn't a bad idea, it's just the hardware wasn't there.

Mahoney: Well, thank you, gentlemen. I appreciate it.

Brackett: I hope it's been helpful.

Mahoney: I think it has.

[In February 2007, Douglas Ross passed away and was unable to edit this transcript]