An Interview with

MIKE MAPLES

OH 387

Conducted by Nathan Ensmenger

on

7 May 2004

Needham, Massachusetts

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

Mike Maples Interview

7 May 2004

Oral History 387

Abstract

After describing his substantial career at IBM where he was involved in display products and then with PCs, Mike Maples talks about joining Microsoft and managing its applications products. He discusses in detail his management philosophy at Microsoft and contrasts it with the IBM approach. He covers Microsoft's successful recruiting practices and how product decisions were made. Maples also describes how development processes evolved and how Microsoft Office was designed and built. The selection of platform focus and decisions on the release of application program interface information are explained. Finally, he details why he left Microsoft and how he did so in a planned and structured fashion.

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
Mike Maples Interview

Nathan Ensmenger: It's May 7, 2004. We are at the Sheraton Needham Hotel in Needham, Massachusetts. My name is Nathan Ensmenger from the University of Pennsylvania and I'm here with Mike Maples.

Please start with a little bit about your background, education and how you got into this industry.

BACKGROUND

Mike Maples: I grew up in a small town in Oklahoma, Orville (?) Oklahoma. I went to the University of Oklahoma and got a Bachelor's degree in Electrical Engineering. Out of school I went to work for IBM. And while working in Oklahoma City as a sales rep, I went to night school and got an MBA in Finance from Oklahoma City University.

Ensmenger: When did you get your undergraduate degree?

Maples: I graduated in January of 1965, and got my MBA in June of 1970.

Ensmenger: When you were in school, did you have any encounters with computers?

Maples: I had a couple. I took a programming course in engineering school on the 1620. In the summer between my junior and senior year, I worked for IBM in the first summer intern program they ever had. One month of that summer program was training on programming. When I went back to school, I went to work for one of IBM's customers, working at home and doing programming for them. They were installing a 1401 and I had developed systems programs for the 1401.

Ensmenger: When you were hired by IBM, what position did you get hired for?

Maples: I was a systems engineer, installing 1400 series higher end systems. After about 9 months (I was in the military, having gone through ROTC in school) I went to training in Maryland and Virginia and ended up at White Sands Missile Range. The first day there I was interviewed by the base commander who had just returned from an IBM customer executive school, and learning that I came from IBM, he put me in the computer directory. It was a totally civilian run computer directory and I was the officer in charge as a second lieutenant because there was nobody else there.

I was there for a year and then I went to Vietnam and I was in an area called Cholan, right outside of Saigon. I was in charge of a computing center -- there were actually four centers I was responsible for all the programming development for the supply of Southeast Asia.

When I first got there we had a small Univac card systems and we had a 250,000 card file we passed every day for inventory items. During the time I was there, I read that France had decided to close its NATO bases and there was an excess computer in France, so I requisitioned the computer and they sent it to us. It was a large 7044 and we contracted with Computer Science Corporation and I became the liaison to Computer Science Corp. to move all the systems to the mainframe.

As I was getting ready for the mainframe -- I was there for a couple months -- the mainframe came and then it was subsequently replaced with a big IBM 360. So that was my early programming experience.

Ensmenger: How long were you in the military?

Maples: Two years.

Ensmenger: And then you came back?

INITIAL IBM ACTIVITIES

Maples: I came back to IBM in Oklahoma. From there I went to the Finance Industry center near Princeton, NJ. From there I went to San Francisco and was the account manager for Bank of America. Then I went back into development and I was a planning manager on the Display products: 2260, 3270 products. I then had a staff job and next I came back as the product manager; at the time I was one of the senior development execs for Displays. At that time in the late 1960s, most terminals were for batch-oriented applications since almost all computers were batch-oriented. Then starting in the late 1960s, people started hooking terminals to them, and by far the most used terminal was the 3270 terminal. So I was there through that rise of interactive computing with the 3270.

Ensmenger: You were in a managerial position at that point?

Maples: Yes; at that time I was what was called a Product Manager. That's the P&L owner for the products. So I had the development people and the marketing people reporting to me.

Ensmenger: As IBM was beginning to expand in electronic computing and to move into new kinds of computing, did you notice tensions or changes within the company? Do you want to say something about IBM and its management practices at that point?

Maples: I don't think there was any change. The management system set up in IBM was that there were a lot of independent departments and a lot of contention among them. The concept was that if everybody looked out for their small piece of the business, the cumulative brainpower on all the issues would help make better decisions.

As the company grew, this process became quite a liability because there were many people who could say no; there were many people who got to review things. So a product manager's job often turned out to be more of a negotiation job with a number of somewhat disinterested parties. But it was much easier for them to say no than it was for them to thoroughly analyze and do what was best for IBM. In fact, they sometimes didn't care about doing what was best for IBM; the system required them to care about what was best for their particular, unique piece of the business.

You had a situation which was compounded with the anti-trust suit that IBM was going through, such that the legal community decided that IBM shouldn't review its competitors' products. So what happened was that the vast majority of competitive activity would be against other IBM products. So you'd be worried more about whether IBM in total might be gaining or losing market share, but you wouldn't even know that. But you would know if you were gaining or losing market share vis a vis other IBM products that were substitutable for you. So you had the lads pitted against each other: Endicott against Poughkeepsie; Rochester against Boca Raton. And that would be the competitive analysis: how well does the System 36 stack up against the Series 1 or the 4300. It was really a very bizarre outcome of a contention system.

Ensmenger: Were there any competitors to the 2260 or 3270?

Maples: There were 2 or 3 classes of competitors. There were substitutable products like typewriter/teletype terminals. There were products that hooked on the lower end of an IBM system like the 5280s. And then there was a whole class of clones to the 3270. There were a number of companies who got in business just by making a cheaper 3270. So, over a long period of time, there were quite a large number of different types of competitors.

Ensmenger: So the period we're talking about now, where you're working with the 2260 and 3270 − what years are those?

Maples: I got out of the service in 1967. I went to Princeton, NJ and was there through 1975 and was in San Francisco from 1975 to 1977. Then I went to Kingston in 1977 and was in Kingston, NY and White Plains and back to Kingston in the early 1980s.

THE IBM PC

Ensmenger: Were you aware at that point of Project Acorn and the development of micro computers within IBM?

Maples: I was in the Terminal Division and so we were worried about IBM trying to do personal computers. I knew a lot about what was going on, but I wasn't involved in it until the first personal computers came out, and then the XT version. Before it came out, I was pretty heavily involved because my organization built the 3270 emulation card for the PC to emulate a 3270. We also built a product called the 3270 PC, which was a PC that was modified with a control program to be a hybrid product between a terminal and a PC. That was an exclusive product sold by the DP [Data Processing] division, not through the resellers.

We spent a lot of time at Boca Raton because we essentially bought the PC XT chassis from Boca, then we added some cards to it, remanufactured part of it, and sold it as another product.

Ensmenger: One of the conventional stories about IBM and the PC is that, as you described, there was competition within IBM about what existing products this was going to compete with? Is that true in your experience of what you know about the PC project?

Maples: There were a number of systems built in IBM to IBM specs. There was a System 32 and 36; but there were a number of systems that were very PC-like, but they were built more to the IBM mainframe spec. And you had the Boca group which only showed disdain for the IBM processes and the rest of IBM.

They went and built a system that was much more based on industry standards and industry technologies − pretty much a back-handed slap to the rest of the business. So there was a lot of tension. When they became very successful, it probably got worse for a while and then everybody wanted to embrace the PC as the big IBM triumph and savior. An interesting reversal.

Ensmenger: In your mind, why do you think they were able to get away with that particular role as an outsider within IBM?

Maples: I think they essentially presented themselves as that. There were a lot of people in the senior executive ranks at IBM who saw the PC and saw Radio Shack and Apple and various folks, and Don Estridge made a proposal to the Management Committee. He said "I can do this in less than a year and this is what I'm going to build."

They told him to go ahead and essentially gave him carte blanche not to listen to anybody and to build that. And his management technique was: he kept charts that he presented in his office and when somebody would come in with a new idea, he'd turn around in his chair and look at the charts and say, "Hmmm, I don't think that idea is on those charts and that's what we're doing."

And so he kept that group very focused on getting that done. Part of the strategy was buying software outside, part was buying standard Intel processors. So he did that and he built quite a good product. Because it was so successful so fast IBM really pumped a lot of capital in and built a big organization quickly.

Ensmenger: So you were still working with the terminal products during the time the PC was emerging?

Maples: Yes, in the very early days. Then I went to what was called the PC Group. But the PC Group had all the intermediate and small systems, including the PC, the Series 1, 36, 38, 4300, and the 8100. It had the industry products, banking and retail products. And it had the printer products and typewriters. I was Director of Development. The role was to deal with IBM standards and IBM development processes and to understand what was going on in the development ranks of each of the different labs.

From there I went into the PC Division and was responsible for PC strategy and business. So I was the person who would negotiate the contracts with Microsoft, or my staff would. I'd put together the strategy and try to coordinate the Boca lab and the Austin lab as to what the software was.

I worked in the headquarters of the PC Division, but I had only the software part. This included the software we developed, the software we jointly developed − primarily Microsoft - and the software that we licensed. We had a number of vendors that we licensed applications from and then resold them under IBM's logo.

PC SOFTWARE

Ensmenger: Was the general strategy within IBM at that point to grow a core business of software through licensing arrangements and other mechanisms, or was there an internal development group at this point for PC-oriented software?

Maples: There were quite large development groups growing up. We're talking about probably 1983 timeframe, 1984; and we're now seeing that the PC is much more relevant to all IBM customers and that it needs to have a much more sophisticated set of software.

So we started the development of OS/2 which was a joint project with the Hursley, England lab, the Boca Raton lab, the Austin lab and Microsoft. We allocated development and innovation funds to those teams. There were several groups in IBM who started dealing with application software: the Office Software groups and other groups started writing software. But we also bought or licensed a lot of software.

Another group tried to get into the application business and licensed a couple graphics programs which were never really successful - IBM didn't really understand how to market PC software. There were a number of other PC products − Software Arts and other people would license products to us.

Ensmenger: One of the common themes that emerges in popular history is the cultural

difference between IBM and lots of the smaller entrepreneurial firms. Do you think that's true and how significant do you think it was as you interacted with companies like Microsoft and Peachtree and other growing firms?

Maples: I think there was a big cultural gap, and it was driven probably by extremes in the process. The IBM process was designed to never have the very largest systems be late - MVS and MVT and the operating systems. The PC software was usually smart guys in a backroom doing anything they wanted as a development process. So IBM was always trying to impose the large system process on the outside developers.

Instead of probably coming to a halfway point and being more of a teacher or a mentor, IBM was more of a dictator: you have to do it my way kind of a role. So, I think there was a cultural conflict. The PC industry didn't think it was necessary to live up to the IBM standards of quality and performance while IBM was trying to make software as good as it did for the big mainframes; The PC guys were just trying to get it out as fast as they could and to try to capture market share.

OS/2 was probably the best example of the difference in development process. There was a very large audit done, led by an IBM fellow. At one time they tried to use an allocation based on the revenues and on the amount of code that was contributed. Microsoft took the position that the quantity of the code wasn't as important as the quality of the code. So there was a big audit to look at the quality of the code.

And I suppose, not to Microsoft's surprise but to IBM's surprise, the IBM auditor came back and said the quality of the Microsoft code was a lot better and it was a lot tighter. And Microsoft really was focused on a small number of really bright people while IBM was focused on a larger number of process-driven people. Not that they weren't bright − it's just that they were living up to a set of standards and a set of conventions that were probably overpowering for the software.

Ensmenger: How about within IBM? IBM had struggled, as much of the mainframe industry did, with process and management and methodology and tightening that up in the 1960s and 1970s. How about applying that within the Company as they're developing their own products. Was that a conflict? Did they attempt to scale down these larger methodologies to smaller projects? Did they try to develop new tools for PC software?

Maples: One of the tasks I chose to do was to try to take what was called the Red Book, which was the development process for IBM, and develop an alternative development process calling it the Quick Path or the Fast Path, so that products that would have had a development cycle of less than 18 months, when you'd usually use 18 months just getting approvals. So products that were deemed less important, or needed to be done fast, could use this Fast Path, which required a lot less review. I developed that process, and then when I went back as Product Manager of 3270, we used that on a number of products.

So there had been attempts to try to modernize smaller product development. The people in Boca at the time − who were running the PC − their solution to the problem was just don't develop any software in IBM; they'd go out and buy everything because the IBM processes were too overburdening.

I went to the PC Division to try to help change that, and our role was to build internal development to a different kind of set of processes that was more like the PC industry. It was infinitely more complex than the PC industry, but it was a lot less complex than the standard IBM process.

So that was the process we were using when we were doing the OS/2 products. It was a very difficult project because of the far-flung nature of the development organizations. You've got development organizations that don't know each other very well, that have different kinds of objectives and that are in different time zones, in different worlds. It's a really hard thing to make any kind of development work. And OS/2 suffered from that.

Ensmenger: Were the IBM developers and managers of those projects people who had experience in other kinds of software, or did you tend to try and hire younger people who wouldn't have had experience in mainframe software development?

JOINING MICROSOFT

Maples: Most of the management group, the architects and the managers, were experienced IBM developers from different places. We did hire a number of young, college graduates. But the young people were trained more in the IBM way than they were in the Microsoft way.

Ensmenger: So you're involved with this range of products including software through the middle of the 1980s, right?

Maples: I left IBM in April of 1988, and went to Microsoft.

Ensmenger: What was your motivation; what did Microsoft hire you to do?

Maples: Here's what happened. IBM and Microsoft put together a road show where we went to about twenty cities. We had a 2 or 3 day seminar on the new IBM PC hardware called PS/2, the new operating system called OS/2, and all the peripheral stuff around that. There were going to be 10 or 12 IBM executives give the keynote presentation. Every presentation started with an IBM and a Microsoft presentation. I gave the first couple of presentations, and as it turned out, most of the IBM executives didn't have the time or inclination to learn the presentation. So I ended up doing most of them and Bill Gates ended up doing most of them. So he and I were together every week for 10 to 15 weeks.

Toward the end of the tour, we were at the presentation before our turn to speak, and he asked, "Would you like to come work at Microsoft?"

I said "You know, I don't think that's a good idea. I'm overpaid now and I love the job I've got. I have a good career here and it's probably just not a good idea for partners to hire each other's folks."

So he said – "Okay." I didn't think any more about it. About three months later, he came and said "I've talked to John Akers and he said that if I had a really good job I ought to offer it to you. I'm sure I can handle the financial situation and make it worth your time. I want you to come to work at Microsoft."

The way Microsoft was organized was that Bill was Chairman, and there were two major divisions. One did systems and one did applications. Steve Ballmer was in charge of Systems and Bill was in charge of Applications. And Bill's time was being pulled away from the Applications to do more CEO activities − manage the sales force, manage international, and so forth. So he hired me to replace him as the head of Applications.

So when he came back the second time I said I'd think about it for two weeks although I had pretty well decided not to go. And my wife said, "Why are you not doing this?" And I got to thinking about it, and I really didn't have a very good reason. And probably, I guess ego took over and I decided that being a big fish in a little pond was better than being a little fish in a big pond. I certainly didn't envision Microsoft dominating IBM. IBM was the power center and Microsoft was a little supplier.

So I went to Microsoft and for a couple of years I dealt with just the Applications business and developed the strategy and reorganized the Application Division. We were relatively successful in building the applications and acquiring market share.

The way the divisions were organized was that they had responsibility for product development, marketing, documentation and quality assurance. The only thing that the two divisions didn't have responsibility for was sales and support. There was a sales organization, both OEM and retailers and then there was a product support organization.

During some of that time Jon Shirley was President and then Mike Hallman came in from Boeing and he was President for a short time but that didn't work out very well. So we decided not to have a President; we reorganized and I took charge of development and marketing; Steve Ballmer took sales and support, and Frank Gaudette had financial and administrative services. The three of us became the Presidents; we were the Office of the President. So there was Bill plus the Office of the President which we affectionately called The Boop, internally.

We would meet 4 hours a week and do presidential things − if there was something that was across the Company or a required agreement or whatever. And the rest of the

week we ran our own businesses, but then for 4 hours a week, we'd play president.

REORGANIZING MICROSOFT

At that time I REorganized the development activities. We had business units which meant that every business unit manager had all the developers, all the testers, all the documentation people and all the marketing for a product family. And I organized it into five divisions. There was the Windows Division which was the operating system division; there was the Database and Analysis Division, which included Access and database products. There was the Application Division which included Word and Excel. There was the Tools Division which was the compilers and all the things needed to develop products. And then there was the Entry Business Unit which was the home products, the multi-media products, the games, all the things that would have been sold more to consumers, not to businesses.

Each of those divisions had a vice president and then within each division there were business units. So in the Office Productivity Division there would be an Excel business unit; there would be a Word business unit and a PowerPoint business unit. Each of the business units would have responsibility for the Mac, for OS/2 and for Windows. So the word processing business unit would be focused on doing word processing for the 3 platforms that we felt were the important platforms, and they would have that set of resources permanently attached to them. So the development guys and the testing guys tried to become experts in word processing, not in computer technologies.

Ensmenger: This suggests a whole range of questions. To begin - your transition from one company to another and how did you deal with a different kind of company, a different structure and culture? It seems to me the biggest change would be the styles of managing product. So in 1988 as you moved from one company to another, what were the biggest challenges or differences that you saw between the two?

Maples: I've often been asked this question. The way I generally answer it is that the most interesting thing wasn't the differences; it was the similarities. Both companies had a lot of very smart people, very focused; they believed they were doing what was right. They thought they were on a mission − that they were going to change the world. The people at IBM and Microsoft all worked really hard. The big difference was IBM was 50 years older than Microsoft. In the IBM lab, the average age might be 45 while at the Microsoft lab, the average age was 19 or 20.

I can remember going to the first Microsoft Company picnic in 1988. There were only two children. Microsoft had 1,800 employees and there were only a couple of them that were married. You had all these young kids who weren't married and were right out of school. IBM had conventional dress codes; Microsoft − it was very much like a college campus. The only difference in how they lived, the hours they kept and the way they dressed, between a college campus and Microsoft was that Microsoft bought the equipment. They all wore their shorts and half of them wore sandals or no shoes. Microsoft bought a lot of T-shirts and things for them. There weren't set

hours − the management system was to let people pick or sign up for what they were going to do, and it was up to them to do it.

So there was very little management attention over directing people or telling them what to do − it was a very empowered work force. And my suspicion was that that was the way IBM was in the 1930s and 1940s. They were much more formal, because the time was much more formal in terms of dress. But in terms of the ages and the attitudes and the mission the people were on − I think it would be a lot the same.

Ensmenger: So what often gets described as a different culture, you see more as a function of the relative maturity of the company in size and practice.

THE DEVELOPMENT PROCESS

Maples: I think the good thing for me, coming into Microsoft, was that it was pretty much a blank slate. There were virtually no development processes; there weren't organizations; there weren't ways of doing things. It was just a bunch of kids struggling to figure out how to do things.

I decided early on that my approach was going to be non-prescriptive. I didn't want to tell them that this was the development process or this is the way you should do things. So what I did with each of the organizations was to say "I don't think it matters what process you use; I just want to make sure you use a process, that it's not a random walk. So every presentation you give me on product status, I want it to start with: Here's the definition of the process we're using, and here's how we're going about it."

I didn't want to start off with coding or with a function. I wanted to know that they were thinking about how to get the product out. So the different business units we had all ran off and started thinking about writing down a process and trying to follow it. And as failures happened - either schedules missed or cost overrun - they'd go back and reexamine. We did a lot of post audits and we examined the process. And over about a year and a half, we had a very common process across the board.

The importance of a common process ultimately was just in the use of terminology. You might have a term like "code complete." But "code complete" to one group might mean when they finished writing the code before they tested it; to another group it might be after it's fully tested. So by developing the process, we codified the terms so that when the documentation people and the testers saw on the schedule "code complete," they all knew exactly what that meant.

About that time, maybe by 1990, we were pretty well into a common process. The advantage of the process did two things: one, it allowed us to build, for the first time, a 3 year plan where I would give them some management objectives – "I want better internationalized products; I want to have the products talk to each other through object linking" or something else.

I'd give them a small number of high level goals. I'd say: "I want you to tell me what releases you're going to have for the next three years. I want you to define what's going to be in them and what will be the focus. Define what's not going to be in them. I want you to have a focus that this version of Excel is going to have a lot better database or this version of Word is going to focus on lawyers. Tell me what the dependencies are that you have on other parts of the Microsoft development organization."

So they'd go off and do that and start thinking about how the pieces fit together with the long term objectives. That was kind of a key benefit of having a process that worked. The other thing was - we were able to be a lot more accurate on schedules. On schedules, we would break the development activities into small chunks. I never wanted to see a larger chunk in an activity than a few hours. I didn't want to see man-days or people-weeks. I wanted to see hours.

My thought there was that if you knew what you were going to do and could define it in some number of hours, you had analyzed the problem very thoroughly instead of broad brushed it. I ended up with huge numbers of tasks − there are tens of thousands of tasks over the development of a process; those would be allocated to different developers and then we would decide if the task was really important to make the schedule. We'd develop a number of schedules, see where we were and redo the functional list.

The goal was to be accurate within 10% of the original schedule. So if someone says it's going to take 12 months, then you had a month leeway of shipping. That way we could deal with consumer products for Mexico at Christmas and the office budgeting cycle of when people bought software in larger corporations. And we were able to get much more rigorous.

The other thing we were able to do with the process was to put a lot more focus on quality. We were able to test better and develop test tools, since we had people who were really professional testers. We put it in a schedule and we were probably − more blind luck than not – quite successful. As the acceptance of the PCs and PC software increased, and more people who had less technical skills started depending on the products, our quality started getting better at the same time. And we were probably a little bit ahead of curve of other companies on that, so that was a competitive advantage.

What we tried to do was to say that development had 3 trade-offs. You had quality, function and schedule, and that quality wasn't an arguable − it was a must. So now you had to choose what function you were going to include or what schedule, but management could only choose one of them. So management would say − this product needs to be shipped in August, and the team would say how much function can I put in before August.

Or, management would say this is the set of functions and the team would come back and say this is the schedule. So the development team owned the schedule, and since the development team included all aspects − the documentation, the manufacturing and everything − they really owned one of the variables and it was their commitment.

The management system I described is on the edge of out of control. By design, we didn't centrally control many things. We let each product team pretty much run alone. And the way that worked was that they would have phases and would report where they stood against that, but we didn't have management sign-off or fixed reviews. The deal was that Bill or I could review any product any time. We could change it or do whatever we wanted with it, but if we didn't choose to review it, they didn't stop and wait. So there would be products that for whatever reason we didn't have time to deal with, that would go from conception to shipping without ever having a management review.

To some extent, all the things I didn't like about IBM I got to change and I probably went overboard on some things. But instead of having to get everybody signed off before you could go forward, at Microsoft it was that you could go forward until somebody asked you to stop.

Ensmenger: In any large or growing organization you need to kind of compromise between independence of the product groups and a federal system that sometimes leads to competition and some overall strategic vision which might have to do with interaction between products or shipping dates - the more mundane things that perhaps the product groups themselves might not have as priorities. Where does that happen? Where do you see your most significant interventions where you'd say to the different product groups: this is a strategic goal for this product that might not be technical.

MICROSOFT OFFICE

Maples: The best example of that stress came with Office. In the beginning we had Word, PowerPoint, Access, Excel − all the pieces of Office were independent products. We had independent product managers, independent schedules, independent design decisions on user interface and so forth. I had come up with a matrix of applications and operating environments, and we chose the Mac and Windows and Presentation Manager, which we later then changed. And we chose a series of 6 or 7 applications: Mail and so forth. The idea was to have a product offering for each environment in each category.

Then I realized that that was a competitive differentiation - that the other competitors didn't have that. And as our product started getting equal to the best of breed, in other words, as Excel was as good 1-2-3 or Word was as good as WordPerfect, this idea of an Office package came together. So I went to Australia and got the Australian subsidiary to bundle all the pieces together and call it Office, and they sold it as a local Australian package. It worked really well in terms of sales.

So then we had to come back to the U.S. and do it broadly in the U.S. The collection of products was less expensive than the individual prices of the products. So now you had a revenue allocation issue. You sell one Office for $500 − who gets how much? And each of the

product guys said. “Well, I'd rather sell full product than sell Office.” Then you had a consistency issue: “I like the way my file menu works versus the way your file menu works.” And then you had a schedule issue: “Mine's going to be ready in November”… “Mine's going to be ready in September.”

So it was the trade-off between what does the Word customer want and what fits into the Office scheme. So we organized. Essentially we took the business units and organized an Office Business unit - we upscaled the organization. But that changed things because the Excel development team was 10-15 people and now the Office development team was 100 or so people. People are making trade-offs.

At the same time, I looked at our products like Word; Version 1 was maybe 30,000 lines of code, Version 2 was 80,000 lines of code and Version 3 was maybe 200,000 lines of code. And you could project that where it's going to be would be 2-5 million lines of code. Our development systems just didn't work in that environment. You talked to the university guys in the late 1980s or early 1990s and the buzzword was, and still to some extent is, object-oriented coding.

So I'd say: “How do you do object-oriented coding?” First you throw away everything you've got. Then you retrain all your programmers and half of them aren't going to be able to be retrained − they're just not going to adjust, and then you start writing everything over again. Having the P&L responsibility for all these products − it didn't seem like exactly a good idea. So I came up with the idea of: how do you disintegrate products from a monolithic big product into building blocks − instead of working from building blocks up, let's work from big products down.

So our earliest project was to take the graphing engine in Excel and make it the graphing engine in PowerPoint and other products, so it would be one graphing engine. We first started with the technology called DDE, which then changed names 2 or 3 times and became object linking embedding. It wasn't academically pure object oriented; it was a way to take a product and make it into sub-assemblies. Then you take that sub-assembly and assign that to somebody and they could take the sub-assembly and break it into components.

We disintegrated products so that certain of them did the Word function, but the spell checker in Word would be used in PowerPoint and used in Excel. And the graphic engine would go across, and the organization chart. So we could buy components and add them to Office, and we could do broader development.

That required some interesting changes in who you made commitments to. The system that we tried to put in place was: the commitment you make to somebody else is more important than the commitments you make to your own team. In other words, if you were doing graphics and you promised graphics to PowerPoint, then you had to deliver to PowerPoint before you did the graphics for your team, which might be Excel. That was a pretty radical shift in the mindset of people, because the whole idea of a business unit was to give them a lot of autonomy. They had spreadsheet customers and spreadsheet functions and they were doing their spreadsheet

functions, and now all of a sudden you're asking a part of that team to honor other commitments, instead of, when Excel got in trouble, throwing off the other guy's commitments.

Ensmenger: Did you still maintain the structure where there was a Word and an Excel and a PowerPoint group or did you break it up?

Maples: Yes to both.

Ensmenger: You didn't break it up functionally?

Maples: No. There were Word, PowerPoint and Excel groups and they owned pieces of common code. We had a group of people who did services for everybody, for example, the installer − so you could install an application on the machine. It was a common piece of code. We had a few common things, but by and large we tried to have every real function owned by a product as part of its business unit, that their primary dependency was their own, but other business units would have dependencies on it also.

Instead of having a graphics business unit, they were part of Excel and the spell checker would be part of Word. When I left in 1995, that's the way it was pretty much organized.

RECRUITING PROGRAMMERS

Ensmenger: Another issue that seems to come up with rapid growth but is somewhat specific to programming is the question of finding programmers. There's this idea that there's something about programming that is not quite learned − that you either have the skill or you don't, and there are all these attempts to find these people and to cultivate them. But there's also just the problem of scale; that it's not a process that scales well, which is why you have these complicated processes and methodologies.

How did Microsoft deal with that problem and how did it change over time? You don't want to lose the fact that things were different in 1988 and 1995.

Maples: In terms of growth, we had a compound growth rate of 87% a year while I was there. We went from $400 million in revenue to over $8 billion. We went from 1,800 or 1,900 employees to 25,000. The development organization went from 250 to 5,000 people. So it was huge growth. The basic strategy we had in most of the areas was: we hired college grads, or we hired new people. We didn't hire experienced people; whether they were college grads or high school grads, we didn't care as much.

We focused our recruiting to the very best schools, so for development we'd go to Cal Tech or MIT, or for marketing guys we'd go to Harvard Business School or Kellogg and we were a little bit elitist in that. At one time I tried to change that, and I went around to all my managers and asked where they went to school. They all went to those schools! We required that all

recruiting be done within the skill of the recruiters. I mean, we didn't have professional recruiters, so the developers recruited developers; the marketing people recruited marketing people.

We had school rules on how to recruit. We required every developer to recruit, and developers would spend up to 15% of their time − usually 8-10% but up to 15% of their time recruiting. They'd go back to the schools they went to. We'd go to a school and let's say we interviewed 100 students. We'd give each of them a 1 hour interview and then decide which ones to invite back. We might invite 10 of them back to Microsoft. For those 10, they'd do 8 interviews, and each interview would have a purpose: we'd test their skills, their resume, their basic intelligence. And we'd have a set of questions that the interviewers would be trained how to do. We were very aggressive. It wasn't: "Let me tell you how good it is to work at Microsoft" It was: "You say you can write in Pascal: here's a problem; write it on the blackboard." Then: "How do you make it faster? And if this condition happened, how would your code handle it?"

So we had a very intensive interview process. The last interviewer was called the hiring manager, but after every interview, as the candidate left the office, the interviewer sent out an email and wrote the result of the interview. So by the end of the 6th interview, the hiring manager had all reviews but the one where the guy was at that time. The first word on the write-up had to be "hire" or "no-hire". There wasn't any "maybe" or... "This guy would be good in some other group". It was ... "I would work with this guy" or not. "I think he would work into my group" or not.

So the hiring manager then had all of these write-ups and all of these decisions or recommendations. And he would decide whether or not to offer the guy a job. If we offered the guy or gal a job, then the next day would be spent in trying to talk them into realizing how good it was to work at Microsoft; that it's a good place to live and all those things…being nice to them for a change.

Of the 10 we'd bring back, we'd hire one. So we hired about 1 out of every 100 that we initially interviewed. So the total picture is: we've interviewed 10 guys, 8 times each - 80 interviews. We interviewed 100 guys one time, 100 interviews − so we've done about 200 interviews for every person we hired. If you think about hours - that's a lot of hours, and recruiters aren't going to do that. Recruiters aren't going to test coding ability. We had recruiters who managed the logistics and managed the reference checks. We also did aggressive reference checks − we had a whole series of questions we'd ask. Most people doing reference checks are trying to be kind and nice and so people aren't as accurate as you'd like them to be. So you come up with a whole set of questions like: "I know this guy is really good, but what is the one thing you'd counsel him to improve?" Or you'd try to get them to reveal what were some of the deficiencies.

Recruiting was the key thing that we did. This is often minimized as one of our attributes but is a key reason why Microsoft was more successful than its competitors - we just hired better people. We just had more, better people. Not that the competitors didn't have some good people. It's just that when you take 500 people, we were just a couple notches higher on average.

Ensmenger: It's apparent that you're spending a lot of time on recruitment, which I can see would have payoffs in other areas. But again, this is not a system that scales well as you're growing. Did that change over time?

Maples: To some extent it's the only way to scale − involving everybody. So if you have 100 people and want to add 80 more, you better have all 100 of them recruiting. And if you have 200 people and you want to add 100 or 150 more, you better have all 200 recruiting. To some extent it's the only way to really stay ahead... you couldn't scale behind recruiters.

At the end we were hiring a few more professional people. I would aggressively target good people. You get to know who the really good people are, and as soon as a company would announce layoffs, I'd send people in but I wouldn't necessarily take the people that were laid off, but I'd call the good people in that company and I'd say " You know your company is downsizing and you may be next." So we were able to pull a lot of really good people out of competitors any time they had problems or something was going on. At the end, we were getting 10,000 resumes a month. Most of them − this is the only time that we ever got recruiters doing things − we'd start each one of them with an hour long phone interview and that would be a recruiter who would just make a judgment on whether to bring this person in to get an interview. We were just overpowered by the large numbers.

We did no "hiring of friends". I'd get a lot of people whose kids were getting out of college and they'd want an "in" to try and get a job. I'd say, "I'll send the resume in to the recruiters" and they'll do exactly the same thing they would if anybody else had sent one in. I'm not going to tell them who to hire. Just because your dad worked there, we weren't going to hire you, unless you were qualified.

We took meritocracy to an extreme. Everybody was equal: there wasn't any advantage of who you knew - it was how you performed.

PROBLEMS AT MICROSOFT

Ensmenger: One of the things that's most interesting to historians, but often doesn't appear in other kinds of accounts is failure. By failure I don't necessarily mean products that didn't work. One reason it's interesting to historians is because history is selective of the things that really worked well. No one wants to talk about failures. So maybe I could just frame that as a general question. It doesn't necessarily have to be products that didn't work out; it could be processes or experience with management. I guess I'm looking for difficulties that you encountered and how you responded.

Maples: Probably the biggest failure in the product category that I was a part of was the product that ultimately became Access. I came in 1988 and the product was six months from shipping and it didn't ship for 5 years. It had a lot of problems. I should have killed the product and re-started it. It was called Omega initially. We reoriented it 6 or 7 times, and ultimately we hired a guy from outside and a couple of others to help redesign it and really start over. We ended up

shipping it in the 1993 or 1994 timeframe, but it was a huge waste of resource and activity.

Ensmenger: My understanding of Access is that it originated with FoxBASE, an outside project.

Maples: No. We had an in-process product inside the company that we were developing called Omega, and it was very far along when I arrived in 1988. It just was a bad product. During that time one of the competitors gave us a very difficult time − Borland. Borland was buying dBase and we decided that because of our inability to get Access out, and because Borland was able to charge a lot for their database products, they could discount very heavily their products that came after us. So they had this cash-cow base that was supporting them. Borland introduced this concept of a competitive upgrade. In the industry you'd sell your product and then you'd come out with upgrades, but only your customers could get the really low upgrade price.

Well, Borland came out with a concept that said − if you own Excel or 1-2-3, we'll give you an upgrade price to switch to Borland. You won't have to buy at full package price. So they came in at a very low price and eliminated cost as a barrier. And they were funding that off Paradox and dBase. So I talked with the guy who owned Fox Software, a guy named Fulton and we ended up buying Fox and bringing that in.

Then shortly after we finished that acquisition, we also brought out Access so Microsoft had both products: it had FoxPro and Access and they were, to some extent, competitive. FoxPro is oriented a little bit more to the developer who wants to build applications and resell them. Access is a slightly more oriented to the end user. But they're comparable, competitive products. They do share some common technology now, but both of them have very different packaging and different customer bases.

WORKING WITH COMPETITORS

Ensmenger: One thing that came up in our discussion of Lotus was their relationship with add-on or add-in products. At first they saw them as competitive and then they cultivated that relationship. Could you say something about how Microsoft dealt with that? I'm wondering how they perceived those companies, how they might have worked with them. Second, are Microsoft development tools seen as tools for generating revenue or tools for creating a whole set of developers who will integrate well with Microsoft?

Maples: Microsoft was always pretty open. We developed macro languages early, to automate the functions that an application would do. It turned out that the macro languages were really powerful because they were really excellent for debugging since they automatically debugged a lot of things. Macros became a very important function for us in our development process for testing.

One of the 3 year goals that I came up with related to each macro language

diverting into a set of things. The language group had built Visual Basic. Each product was also cultivating a group of developers to do add-ons. I don't know if you've talked with Dan Fylstra who was the man who sold VisiCalc - he became an add-on creator for Excel. We bought some statistical analysis products from him for Excel and he became a Microsoft developer.

Then one thing we decided to do was take the Visual Basic product and make it a common macro language across all the products. That did a couple things: one, it gave us a better language or at least a more correct syntactical language. But more importantly, it allowed pieces of applications to be used for a common solution. So if you were a lawyer and you wanted to put together a system that did documents and billing as well as a number of things where you might need to collect some data on Excel and some documents in Word, you could write a set of macros that would span both of them and integrate the functions of both into an application base.

We always encouraged the developers to do templates and add-ons. A few of the add-ons we would license - usually for a year or two. Often if we had a deficiency in the product we would bundle in some of the add-ons that solved that, but we would ultimately re-do that function internally. I don't believe that the add-in business was ever very profitable. If you were a one-man show, you could make yourself a living; if you were a company, you probably couldn't.

The way the system has evolved through my time there in the early half of 1990s, was that we focused the development activity on using Office as the underlying code for a corporate kind of application. So instead of having an application developer or an ISP developer, it would be some corporation that would develop their payroll system or their rental car system and they would just pull in the parts of Office as they could.

So we did things such as: we'd have a contest and give awards at Comdex for the most innovative applications. And we'd find 6 or 8 or 10 applications and demo them all at Comdex and give an award to somebody for the best solution using the Office pieces. I think that's probably one thing where the Microsoft products had a competitive advantage over the others − it was robust enough that the people thought they could really write applications.

Ensmenger: This leads me to another set of questions about technological systems. Historians of technology rarely like to focus on a single technology or decisions about technology because technologies are always embedded in larger systems. Some of them are other technologies; some of them are investments in training and so on. I used to work for a company that did software development, and our decisions about what languages to use, what products to use, were complex. They were never just “this has X features” or “that has Y.” We often ended up using Microsoft for a variety of reasons. The range of companies that are using Microsoft in interesting ways, developing internal applications that they'd never sell as a package, but which have become crucial to the way their internal systems work, suggests that what Microsoft has done very well is to become part of so many other technological systems that it's difficult to replace.

MAINTAINING COMPATITILITY

It also means that in a sense you're no longer managing just the internal company, so you can control recruitment and management practices within Microsoft very well and set your own agenda and culture. But you've also got to think about these larger systems: all the things that are dependent on you for system compatibility. This brings you out into a larger arena and it suggests questions about how you manage that interaction and ultimately something you mentioned earlier: one of the components of these systems is the core and challenges Microsoft's role. How did you manage this increasing amount of interaction with people that you couldn't control?

Maples: The first point is that in your own self-interest you're trying to build barriers to entry that keep your products installed and keep other products out. We learned that because we were originally the outsider. Lotus 1-2-3 had the spreadsheet business; WordPerfect had the word processing business. We were in 2nd or 3rd or 4th place in every product category. So you start realizing what it is they've done that makes it hard to move to you: file formats and compatibility with their macros. So part of your product in the early days has to eliminate those barriers. I'd like to come to you and say, "My product will take advantage of everything you know; it will take advantage of your training; I have a keyboard layout that emulates what you are already using. It'll take advantage of your files, your graphs. You just put my product in and everything works fine."

So we built a bunch of those capabilities; in fact, if you start Word, you still get the option of using the WordPerfect keyboard layout. If you start Excel, you can still run 1-2-3 macros. So then you say: "Well, if I can do it to them; they can do it to me and there's no such thing as a proprietary format." You can say, "If I change it frequently, then it makes it harder for them to keep up." But you soon realize that changing it frequently really addresses this compatibility issue backwards. Your customers who depend on your functions need to be able to run that same set of functions in the next release or the next version.

So you start thinking about what are the barriers to entry that I can do and how do I manage them? And how long do I want to keep them as a barrier? For example, the Visual Basic program which we developed across all of our applications. There comes a time when it becomes important for that to become an industry standard, not a competitive advantage, not a barrier. There comes a time when your Office collection would profit from somebody else's payroll, accounting or something else. If they had the same programming language it would integrate into your application.

So it's an issue to manage as to when you bring it in as a competitive advantage and how well and how fast you spread it across your products. How long do you let it stay as a competitive advantage and then how do you market it once it gets customer acceptance and usage? How do you market it to other people so they can embed it?

At that time you would have no legal recourse if Lotus 1-2-3 wants to use your language. From a competitive point of view, your timing questions boils down to: if I come out with a language and customers start requiring a language, Lotus will probably write a language. At that

time I will release my language to other people who will want to integrate it, and there's a good chance the competitors will not be far enough along with their own not-invented-here kind of language, that they'd come and use the one that I put in the marketplace. So it's just a question of timing - of building the barriers to make it difficult to move from you, and then timing it − making those industry standards except for your competitors.

Then the last component is that once you have something that's pretty well entrenched, how do you provide for your own upgrades. Because your competitor is your own previous release. If I have 100% of the business, then I am the only competitor I have; well, in the early days, the word processing competitor was a pencil or a typewriter. Then it became other products, and then it became that Microsoft's competitor for Word is their last version of Word. So you want to be able to deal with that last competitor you had and migrate their macros, their applications, and still bring in new functions. And the new functions may require some file changes, so you've got to make sure that the new files you create are at least read and acted on by the old versions as well as the old files acted on by the new version. Because you're not going to change everybody on Day One, it's a closed-loop process: I am both a creator and a user of documents. And whether I'm the old guy or the new guy, I'm still both.

So you have to spend a lot more time thinking about that. That's the reason organizations get a lot bigger and they use a lot more planning people.

PRODUCT DECISION MAKING

Ensmenger: How do those kinds of decisions get made on a practical level? Who made them and how do you make them since so it's not just an internal decision, you're in a sense coordinating lots of players as well.

Maples: Here's what you do: you say, "It's important for us to build barriers of entry." You make sure that people understand the economic theories, and then when you don't like a 3 year plan or a 5 year plan, you say, "Here are the things that I want to emphasize − the small number of things I want to emphasize". And then the team owns the problem. I might get angry at them or might criticize them for not doing well enough, but it becomes a customer requirement that they had to deal with and prioritize among other customer requirements.

You can say, "Well, I can either make the file compatible or I can add this new table function". You need to decide that and you need to figure out which customers it affects and how it affects them. They're your customers, it's your product and you need to make it successful.

But the process of choosing functions and features is a kind of triage process where a lot of people get to vote. The market people, the developers, and Gates or I or the product team can get in and vote. So I'll have my favorite features and Bill will have a couple of favorite features and Pete Higgins will have some and the development team has some. You get into these long meetings where you rank all the functions. First you start off and say: "A spreadsheet does

analytics, graphics, database functions, 10 things − and we're only going to focus on 3 of those this time". So when you start off, we're going to worry about analytics. You line up all the analytics features and then you say, "Okay, these are all the things we could do and these are the priorities." Then you say, "Okay, developers, we're going to ship this product on August 31. How many of these features will you commit to do?" So they go off and write up the development plan and say, "We can do the top 8".

So we say, "Okay, for checkpoint 1, we want to do 1, 2 and 3. And we want you to code them, finish and test them and come back". And Checkpoint 1 is six weeks out. So six weeks later we look and say, "Where are you?" They say, "Well, we were late and only got 2 of them done." So we say, "Okay, finish the third one" and then we have to decide: we can't do all 8. We're only going to do 6 of them.

Ensmenger: With so many different parties, it seems at some point there might be a shift in emphasis. Early on in a product's life, the technical group might have set the Word document format which ultimately becomes so entrenched that the marketing and sales support could rationally have a much greater say about compatibility issues than new features. At some point there's a shift in priorities, and how do you deal with that? I understand it's ceding lots of power to the product group, but at some point you or Bill Gates has to do that.

Maples: We try to never do that. We almost tried to have the Japanese system where everybody argues their case and everybody ultimately agrees what's best for the product. I was at Microsoft for 8 years and only twice did Bill tell me what to do. One time we were debating Visual Basic and he wanted to have user outside vendor constructed controls . The product was nearly finished and I wanted to ship the product. We argued about it for maybe a week, had two meetings a week arguing our case. Neither one of us changed our mind. At the end of the time, Bill looked at me and said, "Look, we're going to put them in because I'm Chairman." I said "That's fine - that's your job and I understand that - why didn't you tell me a week ago!" So we put them in and he was right; it was the right thing to do and we deferred the product for 3 months.

The other time had to do with how much word processing function do we put in Windows, in WordPad for free. We had Works; we had Word; and we had WordPad that came with Windows. So how functionally rich should we make WordPad? The Windows team wanted it to be as good as Works or Word and give it away. And I wanted to sell Works and Word. So we had a big argument and finally Gates said, "We're going to do it this way because I'm Chairman." Essentially they decided to do a bunch of functions and I told him I wasn't going to do it for him. So they went out and contracted it; the contractors didn't do a good job, so WordPad's pretty irrelevant. So Bill won but lost on that one.

But in 8 years those were the only two times that we didn't come to an agreement, where somebody arguing their case didn't convince the other one to change. And you try to do that at the lowest level you can. You try not to have a top-down decision. The team who's building Word knows more about that tradeoff than I will ever know or Bill will ever know. So if it should never

get to where you have to make that decision.

Now, when you review a product and look at things, you see that just doesn't work right or it's crummy, then we'll often re-work it. It's not as if you were arbitrating a feature. You're just saying that it's not acceptable for a Microsoft named product. It loads too slowly or how can you expect your user to do that, or, if I'm a user, am I going to sit here and do this? So you challenge people on that, and require them to make changes. But by and large, the triage of what features you do we tried to have them decided at the lowest level. The marketing guys would argue with the developers and they'd argue with the testers and the sales people.

APPLICATION PROGRAM INTERFACES

Ensmenger: The earliest structure you described as you arrived in 1988 divided Systems and Applications and you were in Applications and Steve Ballmer was in Systems. Later you described a slightly different structure that was a little more functionally oriented. This relationship between Systems and Applications is obviously complicated for lots of reasons, technical as well as legal, and can you speak to that

Maples: When I came, the deal I had with Bill is - I wouldn't work in Systems for two years because I had worked on systems at IBM. I knew all about the IBM strategies and all about their products and so forth. So just to avoid a conflict of interest, I didn't want to be the other side of the table from the people who used to work for me.

So for two years I did Applications. Then when we reorganized I had all the systems and applications and everything together. But this whole firestorm I started almost by accident. I don't even remember how it happened. I was interviewing with somebody, some press person, and I mentioned something about APIs [Application Programming Interfaces]. Then someone went off and wrote a book about it and it became a real firestorm. It was a non-issue really − it's just that it sounded bad. And the issue is − when you write an application, the modules talk to each other. Sometimes you want those modules to talk to each other in a fixed way over a large number of releases, so you publish that as an API which is your guarantee to the world. It's going to be supported in each release; they won't have to rewrite the code.

That's both for ISVs [Independent Software Vendors] and for corporate customers that write code using your products. The other APIs depend on the way that modules are organized and you know they will change over time. You know that these modules will be combined, they'll be done away with and there will be something replacing them. So the way they communicate isn't defined to be persistent. So you just don't publish those APIs; but programmers will use them. Virtually every programming group will take a debugger and look at those and figure out what they are. If one of them is changing the date from a military date to a regular date and there's a way to do that easily, they'll see that and figure it out and use it. They'll decide to take the risk as to what they will do if that changes in the next release.

There were some applications programmers who had huge advantages because of that. So to quell that, we decided that we would publish every API that we could find and we started a developer network. You could buy a developer kit that lists 10,000 APIs or so. It really didn't change anything. The big competitive advantage that our competitors never dealt with was the fact that we had a compiler technology. I had a group in the application area that wrote compilers and we didn't use any industry standard compilers. We didn't use Microsoft compilers or anybody else's. The compilers − we generated our code into a pseudo code called P code, and then we built an interpreter that would run on the Macintosh and on Windows and on presentation matrix. So the same basic code that you had for Excel could quickly be moved to the Mac. But the real advantage was that the code size was dramatically smaller, maybe a fourth the size. So we represented function differently than you would in a normal C program.

That allowed us to have, in the early days of the Macintosh, a competitive advantage in graphic applications because we were there early and successful on the Macintosh, which was a minimum memory, minimum system in the early days. So we had to write code that was small and efficient for a graphical system. We had a lot of customer feedback about what was a good graphical system.

So when Windows finally got to where it was moderately successful and OS/2 was moderately successful, we had a way to pack a lot more function into smaller memory spaces. That allowed our products to run on smaller systems, to run on smaller memory, to perform better. That was one of those competitive advantages we never released as a product. Some tools you want to make products. Ultimately before I left in the mid-1990s, we converted to Microsoft programming languages. We established a set of functions that we required; we gave it to developers just as if we were a customer and said, “If you build it this way, we will come”.

They ended up putting the code that we wanted into the regular compilers. The applications guys wanted the regular compilers, and then they converted. because there comes a time when the functions and features of the tool are growing so fast that you end up putting too many people on it.

But that was probably a lot more of an advantage to us than the internal calls. Internal calls - I can't think of any real competitive advantage that they created, but it was a big firestorm. And still the lawyers ask me about it all the time.

PLATFORM FOCUS

Ensmenger: I can imagine. Early on Microsoft was working with lots of different hardware and platforms and vendors. Early on it was more deliberately targeting lots of different platforms. When did that strategy change? Is it just with Windows? Does that have structural implications or is it technical?

Maples: In the early days of the PC, pre-IBM or IBM's entry, every architecture was

unique. The Apple II was different from the Apple III. The Apple was different from the IBM PC, which was different from the TI or the DEC, or the Wang or any of the other Intel-based PCs. And all of them were resource-constrained. Memory cost a lot of money; everything cost a lot of money, so you had to be very efficient.

So Microsoft's strategy in the early days would be to be able to take the word processor and run it on a lot of different architectures. It turned out that the world very quickly adopted the IBM standard as the standard PC - the Intel-standard PC. And the IBM PC became the reference platform. IBM published the interfaces and all the specs, so it was relatively easy to copy. So the previous Microsoft strategy would not have had any competitive advantage.

In the first few years of the IBM PC − 1982-1984 − Microsoft's applications were relatively insignificant. They never became the winners. About that time Microsoft also invested in the Macintosh and wrote Macintosh applications. And in virtually every category Microsoft dominated that category: word processing, spreadsheets. It did not have a product in the database arena. So in doing that they learned about the value of graphical applications, the value of an operating system, the value of a number of things and got a lot of customer feedback.

So Windows comes along which was hard to write to. Microsoft modifies the application for Windows. PM comes along, OS/2 program manager, graphical user interface - Microsoft takes its applications there. And we just had the experience of better graphics and we were committed to graphics. We were in the market of character-based applications but weren't significant. So we had nothing to lose going to graphics. Our competitors had an installed base to lose. They had to worry about compatibility - a whole bunch of issues. We were out telling everybody that there ought to be graphics; don't worry about the past.

So we probably had a year or two, or maybe even three, of competitive advantage before they were willing to change and had the skill to change. The second thing that happened is that they saw Microsoft doing well in the operating system business and they saw these other businesses and so in order not to fund one business with another, they preferred that OS/2 win versus Windows. So they made decisions to exclude one or the other.

Microsoft was there with both Windows and OS/2 applications. IBM, WordPerfect, Lotus were saying − we're only going to do OS/2 applications. All of the upstarts were doing Windows. It required less resources; the customers were embracing it. IBM finally decided that the right strategy for OS/2 is to say: “We will embrace Windows and in OS/2 you can run DOS and Windows apps and OS/2 apps.” In fact, they licensed Windows to embed in OS/2.

As a developer I'm thinking, “Why am I spending 10% or 15% delta cost to make OS/2 version when I've got a Windows version and they're saving us all the work?” So virtually every vendor did that except for the major players. So you still have the WordPerfect guys and the Lotus guys - a few people who are still saying that OS/2 is going to win, but OS/2 just never won. So they just lost share before they could recover and get their products to market; they went

downhill.

It was an emotional time. Microsoft's technique for dealing with a vendor's emotion is to get them riled up to the point that they make emotional decisions instead of business decisions. And once they start making emotional decisions, they make bad decisions. If you can keep agitating them and goading them and leading them on, then you can get them to make bad decisions and then you can win.

Ensmenger: Microsoft has moved from its operating systems and languages to applications and has had great success. It has continued to expand its range, most famously the way it dealt with the emergence of the Internet. Did you think about the limits of what Microsoft wanted to do? There were other niche markets that it could get into if it chose to − some of them I'm sure are financial decisions − but there are other boundaries that are a little more blurry. Hardware is one of those boundaries. How do you think about that in strategic terms, especially in this pre-Windows era?

Maples: I think we had two guiding principles. One was: we felt we had the right to go anywhere there was money to be made. The second guiding principle was: we reserved the right to change our mind. So we didn't want to have proclamations, you know? We are NEVER going to do this or that, because you don't know what you don't know.

Our strategy was relatively easy to understand and a lot of people still hardly believe that that was the strategy, but we said that we don't want to be in the applications business except for high volume/productivity apps. We want to be in the tools business and the platform business. We want to take our high volume applications, make them ubiquitous and make them a tools platform also. We don't want to do accounting; we don't want to do payroll or most any kind of real business application. We want to supply tools that the user can develop it or we will encourage the industry to develop it.

Probably the only place that that's still not true is in what I call small intermediate business. Let's talk about hardware. About 30% of hardware goes to large businesses. About 30% goes to homes and consumers, and about 35% goes to small intermediate businesses. That's just hardware sales: computers, printers, etc. You look at software sales. 95% go to the home or to the big business; 5% goes to small intermediate. So you say the small intermediate business is either profiled differently than the bigger businesses or they steal more software or something happens different.

When you talk to those users, they're not like a corporation where people are doing word processing and a lot of stuff on the machine. You go look at a cleaners or a sporting goods store, and they're doing basic accounting. They're buying QuickBooks or Peachtree or something like that, and that's really how they use their computer. Or they buy specialized applications − hardware guys have these machines that can print letters on Jerseys or stitch logos on ball caps. Or they'll get a machine that comes from their wholesaler − you belong to a buying

syndicate. They put a machine in so you can do orders with them or a machine that does Federal Express or UPS or whatever.

So the machines became much more specialized. So Microsoft sits back and says, "Okay, if I'm going to get more dollars of revenue out of that segment − I don't have the hardware − I need to figure out how to do what they do." That gets you into buying Peachtree or doing applications for the small intermediate business. And that's just because that customer set is not nearly as productivity or database or high-end oriented. But by and large, its operating systems, infrastructure like Sequel Server and Mail and things like that are productivity apps and consumer apps. That's what Microsoft has been since the mid-1980s.

It's probably reasonably easy to pick out places to go that Microsoft won't go, but Microsoft's not going to say they're never going to do payroll.

ACQUISITIONS

Ensmenger: One of the reasons for expanding − perhaps it's financial or expertise. But one mechanism is acquisitions and you mentioned Fox and that relationship. Can you think of other successful or less successful acquisitions that Microsoft has made and why do you think some worked and some did not?

Maples: Acquisitions are hard to make work. While I was there, we bought 13 companies. The best known was PowerPoint from a company called Forethought. We bought the Mail system from Network Courier. We bought FoxPro. We approached the business quite a bit differently from purchasing a product.

We never thought about buying customers nor revenue streams - we always thought about buying people and skill. You would decide that you needed to be in the mail business and it would take some amount of time to build the skills and the products. Then you'd go out and look at who was the mail vendor you wanted to buy. You would end up choosing to buy somebody based on their skill and ability, not on their products. The reason Microsoft almost always did that was that a product has about a 2 or 3 year life. And so if you're buying a 2 or 3 year life product and everybody bails out on you, you really aren't buying very much. It's pretty expensive growth.

We always wanted to figure out how to get the people. In the early days, virtually every acquisition was moved to Redmond. So we would buy the company; we'd move all the employees to Redmond and then we'd take about half a development team and put them into other Microsoft areas and about half Microsoft people into their areas. And we'd try to integrate them very fast.

The first one we didn't do that with was PowerPoint, and it's still in the Bay area. The second one we didn't do that with was Mail - it was a Canadian company in Vancouver. We left it there for about a year and then moved it. FoxPro we moved the day we bought it. Most of them

we moved. We always thought that growth was through getting good people more than through buying products or customers. I think that's probably different from most acquisition strategies.

After I left, Front Page was bought and a number of companies and products were bought. There were some that didn't move and most of them did not work out well. Web TV − the people who started it all left. I think that's the worst case scenario: you buy a product that's 70% there; the market is just developing and all the smart guys leave.

MANAGEMENT SELECTION

Ensmenger: So that's about acquiring products and product groups and developers. How about managers? In some companies there's perceived tension between the managers and the developers and the managers and marketing. Microsoft seems slightly exceptional. But you yourself are an import from another company. You mentioned you'd hire developers from Cal Tech and MIT, but that you were hiring managers from the Sloan School and Wharton.

Maples: We hired marketing people from Wharton.

Ensmenger: That was my question − where did management come from?

Maples: Management was always grown from within. We had a very large span of control − 15 or 20 people. So we were not very well managed, but managers were all pretty much grown from within.

Ensmenger: Did that continue to be true over your tenure there?

Maples: Yes, and I think that's true now. Microsoft always started with bringing in some senior managers and I'd say 70% of them didn't work out. But the vast majority of those that are there now are the ones that grew up in the business. The vast majority of the first and second line managers either came with an acquisition, or were there, or were grown from within.

Management development wasn't one of our strengths during that period. It was kind of a sink or swim attitude. It wasn't till the end of my tenure that we really started focusing on how you train managers to be better managers and how you develop the managers you have to be senior managers. When you're growing so fast, you just hang around for two years and you're a senior guy! You're there two years and you're in the bottom 20% of the employees.

SUMMARY

Ensmenger: Are there any concluding remarks you'd like to make on why you left Microsoft and what you're doing now?

Maples: I had absolutely the best job in the world. I got to work with really smart people.

We were building great products and things were working very well. It was absolutely the best job in the world.

In early 1994 to mid 1994, we did our personnel reviews and Bill would give my review to me. The way the personnel review system worked at Microsoft is that the employee would write the review of what their objectives were and how well they thought they did, and then the manager would add to that.

Bill was always very thoughtful and always spent a lot of time on the reviews. It would be a 3 or 4 hour meeting and he'd spend a lot of time before that. Frank Gaudette died of cancer and my wife and I were at the funeral. There was some comment about how much he loved his kids and how much time he spent with them, but I knew he was working like me − 15 hours a day, 6 or 7 days a week.

I had made a list of things I wanted to do before I died: go to Africa, Alaska, China − a number of things. My wife and I had had very few vacations. I'd had 4 vacations in 20 years or so. So I knew I had really short-changed her. So we decided that in 1994 − Windows 95 was about to ship, Office 95 was about to ship − that we'd finish those products and if I stayed, then we would be working on the Internet strategy and it was going to be a 3-year release cycle for the next set of products. So if I was going to stay I needed to stay from 1995 through 1998.

So I looked at my age and my health, and I said, "If I'm not careful, I'm going to be doing these life tasks in a wheel chair or something's going to be seriously wrong, and I really need to think about doing them and being more healthy."

I wrote my review, and instead of writing a normal review I started off and said, "This will be my last year at Microsoft. Instead of talking about what I did the last year, my effectiveness and so forth, what I ought to do is lay out a strategy for what we should do to prepare for me leaving." I described whom I thought would be the right people and while we'd try to bring some people from the outside I said I thought from a long-term strategy Microsoft should promote from within and see if we could reorganize and make it work.

So Bill comes into the review, and it's the first he's seen the review. He'd been busy and hadn't done anything in advance. I sent it over in an envelope. He opens the envelope and reads the first sentence. He just kind of said, "WHAT?! What are you saying! You traitor!" So he goes through an emotional period. The first week or two it's, "You traitor! You no good son of a bitch." Then he goes through 2 or 3 or 4 weeks of, "I can buy you. I just have got to find out what the right price is. If I give you enough money, enough something, I can talk you into staying."

Then after about 2 months, he comes to the conclusion that I'm really leaving! So then we get constructive. Paul Moritz is really a good guy and I put him in a staff job because he needed some things there and then we restructured the organization. When I left, we broke development into three groups. We put somebody in charge of all three groups from within. So I

was pretty happy; it was very smooth. Through that period of 10 months or so, we moved people around and gave them some new experiences. We didn't really talk about it till about 2-3 months before I was leaving − outside of Bill and I. And then we let people know, and so things worked out well.

It was just that I, for personal reasons, needed a change. I had done my 32 years and it was time to leave.

Ensmenger: Good, I think that's an excellent note to end on. Thank you very much.

Maples: I can give you one funny thing that happened. When I was leaving, they were having a hard time thinking of a going away gift, so they end up buying a small herd of fallow deer. The newspaper guy wrote an article about my going away gift and he said, "What do you give retiring Microsoft execs?" Six bucks and a doe!" The implication is dollars.

Ensmenger: So what did you do with this herd of deer?

Maples: I put it on my ranch. I have a ranch in Texas and I raise exotic deer and antelope. I have deer from China, India, Mongolia, Japan, Africa, Europe, Denmark, Poland and the U.S. I have elk and I have antelope. I have 13 species and about 350 animals. It's a hobby. I have 3 endangered species − one of them is extinct in its native range. So that's what I do.

Ensmenger: How's the rest of your list coming along?

Maples: I've done most everything I wanted to do. I've been to Africa and most of the rest.