PC Software Workshop: Technical - Customer Support

Moderator:
Larry Welke

Recorded: May 6, 2004
Needham, Massachusetts

CHM Reference number: X4621.2008

Table of Contents

PC Software: Technical - Customer Support

Conducted by Software History Center – Oral History Project

Abstract: People who were involved in customer support in the PC industry discuss the basic need for support and how they managed the support costs. They talk about the various types of support including documentation, user groups, corporate help desks and instructional wizards and how they tried to reduce the overall cost of support. Finally they talk about how they distributed bug fixes and the role of in-house testers.

Participants:

Name	**Prior Affiliation**
Larry Welke	ICP Directory, moderator
Adam Green	dBase books and seminars
Evan Koblentz	Computer Collector E-Mail Newsletter
Mike Maples	IBM, Microsoft
Burton Grad	Software History Center
Janet Abbate	Virginia Tech, historian
Nathan Ensmenger	University of Pennsylvania, historian
Thomas Haigh	The Haigh Group, historian
Mike Mahoney	Princeton University, historian

Larry Welke: Let me be mundane and ask everybody to announce your name, rank and serial number just for the sake of the tape recording. Nathan, we'll start with you.

Nathan Ensmenger: I'm a historian at the University of Pennsylvania.

Mike Maples: Mike Maples, Austin Texas.

Mike Mahoney: I'm a historian and am at Princeton University.

Adam Green: Adam Green, currently with no company.

Evan Koblentz: Evan Koblentz, Computer Collector E-Mail Newsletter.

Thomas Haigh: My business is run through The Haigh Group and I'm a historian.

Janet Abbate: I'm another historian and I'm at Virginia Tech.

Welke: I guess we'll just start with the questions that the historians might have and take it from there. Janet, why don't you go first?

The Basic Need for Support

Abbate: Sure. When you're developing a product or releasing it, how much support did you think the customers would need? Did you think that it would be like other consumer products where customers get the product's instructions and just kind of go through them, or did you think the customers would need constant support that you or the dealer would have to provide?

Green: I guess the majority of the support is right up front during installation. And it's more a matter of customers wanting to know you're there and then calming down about it. So the support drops off dramatically. And the documentation is one of those things where you have to do it but no one will read it.

Haigh: Adam, can you tell us a little bit about your activities in the 1975 to 1995 period?

Green: Well, I've done a lot of things in software that were related to support. I ran a mail order company for a couple of years, and then I wrote a book. I wrote the first book on the dBase language and wrote a number of books after that on dBase and other products, and then I did training. I went around the country and taught seminars on dBase and other products for about 10 or 12 years. And I wrote a newsletter and magazine columns and things like that.

Welke: Any other comments?

Maples: Well, from a Microsoft perspective, we always knew there would be the need for support. It turned out the majority of support issues were "how to" problems: how to make mail merge work or how to do whatever. If I were to profile the late 1970s, we started charging lots of money for software packages, or what we thought was lots of money at the time – $500 or $600. So there was a view that we ought to include a lot of things for the price. We started packaging the software in fancy boxes and it got to where the cost of manufacturing a product was up to $200 a copy in some cases. And we also thought that the customer needed to get a lot of support.

We did a number of studies about this. Among other things, we found that the need for documentation is very variable by country. In some countries people want to be taught by people. In other countries there's the culture that they are taught by reading. In the United States, the culture is that most people learn by trying. In the U.S., we surveyed the users and over 50 percent of them never take the shrink wrap off the manuals. But on the other hand, there are some people who use them a lot.

The second thing we learned about documentation is that people will go out and buy a $29 book that isn't half as good as the documentation they got for free with the product, and they think that it's a better deal. So in the mid-1980s, we started reducing the product cost and we were able to get the average product cost down between $10 and $20. We recognized that the users cared a lot more about the software than they did about the nice paste board box around it and the three ring binder with the linen cover.

Mahoney: But the size of the box had to stay the same.

Maples: The size of the box had to stay the same to some extent because that's the way the shelves were built at the re-seller.

Managing Support Costs

In terms of documentation, we started going to a more minimal documentation, first because if you're going to translate it into 40 or 50 languages, and that's the gate to releasing a product; you want to have less to do. And we really geared up outside Microsoft-labeled books. We'd publish three or four books and we got lots of other people to publish books and articles on the products to fill in the documentation gaps. We spent a lot more time trying to deal with on line help and on line documentation instead of the printed documentation that got sent with the product.

In terms of product support, the rule we put in place was that every new release of a product had to have half as many product support dollars spent as the previous one. What that meant was that we could have half as many calls, or the calls could take half as long, or some combination. For each 100 units sold, it had to have half as many hours in product support as the amount the previous release required.

Haigh: When did that take place?

Maples: Maybe in the 1986, 1987 time frame. And we did two or three things to manage it. One thing was I required every developer to spend a day each quarter being a product support person on the phone to hear what kind of problems customers had. The second thing was we got reports that categorized the product support calls by type of problem and other details. And we would try to eliminate the top ten problems on every subsequent release. If you can eliminate the top ten problems, you generally can cut your support cost in more than half because half of them are about some little thing that everybody calls about.

Quite often I would go over to the support area and just sit around and listen on the phones or talk to people. While doing this one time, I went by a word processing product support person who had a couch behind his little cubicle. I asked him why it was there. He said, "It's my mail merge couch. When I get a call in mail merge, I know exactly what the problem is and it's going to take me 30 minutes to walk the customer through it. I just get up and lie down on the couch and walk the customer through the problem." As a result, I thought, "We ought to fix mail merge."

So we worked on mail merge and fixed it some. And all of a sudden, the number of users went up by a factor of ten. So now we had ten times more users and even though we had one tenth the number of problems, the result was that we still got the same number of calls that we got before. So you're always in that game of trying to knock off the biggest problems, and in many cases you just create the next level because you've simplified something to where a lot more people can use it. I'm assuming that Microsoft is still doing the same thing. Support is such an important driver of your expenses that as your volumes go up from a few hundred thousand to millions, you just can't have a linear increase in the number of your support people.

The other trend was to start charging for support. There are certain products that charge and some that don't – for some of them, customers feel that support should be included in the price and for others, they don't.

Abbate: When did Microsoft start charging for support?

Maples: Probably before 1985. I think Windows and DOS have charged for support for a very long time. Customers get free support for a 90 day installation period to help them if they have a problem with installation. But we got overwhelmed with "how to" calls. The second thing that happened with Windows and DOS is that they were sold primarily through OEMs. Part of the reason for the OEM discount was that they were supposed to provide first line support and service. But most of the calls we got were from people who were trying to get around that. They called Dell and they didn't get a good answer, so now they're calling Microsoft. We told them, "You can get support from Microsoft. You just have to pay for it. Or you can go to your hardware manufacturer and get it for free."

Ensmenger: That point came up in the accounting session, and I was quite surprised to hear that many of those firms always charged for support; it was built into their profit model. And some of them indicated that it was a pretty assured revenue stream. I think that has to do with who their clients were; that often they were selling to the people who might otherwise have gone to a CPA or a consultant or another kind of service for help. And so there is the expectation that they pay for service in a way that someone buying an individual copy of Microsoft Word has not.

Maples: And I think that the issue there is “how-to” questions versus “break-fix” questions: for example, how do I make mail merge work or merge an Excel sheet with something else versus this thing doesn't work right or it's got a bug. And I think most vendors have kind of tried to take a position that how-to questions should be paid for or you should figure that out on your own, and break-fix questions are ones that you should tell me about, and I'll try to figure out how to do a patch or something. And I think the recent history on break-fix is you go to the web and you look in the database and you find out that in almost every case your problem has been identified earlier.

Welke: How did the change in the nature of the market that resulted from the tremendous acceptance and growth of the PC affect the kind of requirements that you had for support and documentation, especially because most people won’t read the documentation. How many companies anticipated that growth to begin with and worked that into their strategy?

Maples: Well, I think the strategy is that the need for support is the result of software problems. It's not a documentation problem; it's not a support problem; it's a software problem. And if I build my software easy enough to use and functionally rich enough, I won't get a lot of support calls. So the solution to the problem is better software, not better documentation or more support people. This means that you need to do a lot of things to influence the development process to reduce the need for support. If you could, you'd get away with no support and no documentation. When you think about it, for most things that you buy, you don’t get much documentation. For example, if you buy an automobile for $68,000, they give you a little bitty book and you don't even read that. Because it's easy enough to use, you understand what you're doing, and you don’t need to learn about it. But by and large, it's a development problem that you have support calls and it's a development solution to eliminate them, not a support call center solution, because the support people can't solve the problem, they can only get more efficient. They can't make the printers work better and they can't do other things.

Haigh: Now it seems there is a system where you might have three or four different levels of support, starting with people who can just answer very simple kinds of top-ten questions, with escalation up to the development team for sufficiently esoteric problems. When did that kind of tiered support system begin to emerge?

Maples: Well, I think in the early days there was a lot more development activity in the support lines. At that stage, I think the support calls were a lot more about problems with the program – you know, the product doesn't work right – and since the support people didn't have the ability to fix it, the call went to the development people a lot faster. Today there is a formal way of getting information from support to development, but it doesn't happen very much. In virtually every company I know, the product support guys build knowledge databases of issues. When a support call comes in, the first half of the call is profiling the customer – what's your name, what kind of machine do you have, what release of the operating system do you have, what version of the processor do you have. So 50 percent of your cost is in just finding out the characteristics of the customer. So there is a good case to be made for automating the front end to identify that information before you even talk to the customer about the specific problem. The next 25 or 30 percent of the time is spent looking in the knowledge base, because a very high percentage of the time – let me say 97 percent to 99 percent of the time – the particular problem has been reported and there is a known solution to it. So now you're down to 1 to 3 percent of the time where it’s a new problem and there is a need to figure out exactly what's going on. And if I ultimately determine it's probably a user error, I'd just tell them to reformat their disk and start over.

But if it's a problem that looks like it's a technical problem with the product, then I'll create a new entry in the knowledge base. I'll describe it as best I can. I will determine if it's repeatable or not because problems are almost impossible to solve unless you can repeat them so the developer can set up the situation and repeat it. Then this information would be given to the development team to try to figure out what to do; either to produce a patch to fix it, find a workaround, or something else.

Haigh: And do you know at what point those electronic knowledge bases were first being developed?

Maples: It was before 1985. They were certainly in a lot simpler form then, but as long as I can remember there's been somebody logging problems into a database.

The Role of Gender in Support

Abbate: I’d like to look at the gender question here because you made comments about how people don't read documentation. But I think women actually do read documentation, even our car manuals, for that matter. I think men want to just put their heads into the machine and just mess around with it, and women want to read how it works first. That's been well researched. But I'm also wondering, are women more represented in support than in development? And was that true with Microsoft?

Maples: In Microsoft, I think that the documentation group was more women than men, and the support and the development teams had more men than women. I can't speak to how it is now. Microsoft had a reasonably high percentage of women, something like 30 or 40 percent.

Abbate: Did you address the gender issue directly?

Maples: I was worried about it from a development point of view. Microsoft has a reputation of being a very tough culture where people are very confrontational and argue, and women felt intimidated by it to some extent. Because of this, I always was very interested in trying to get the women together and figure out what we could do about that. So I met with the women, particularly the development managers.

Mahoney: I was thinking more in terms of the user – male versus female users – and their different needs and different patterns of support.

Maples: I don't remember if that had ever really been a question.

Mahoney: And was there a difference in how you talk to a caller, depending on whether it's a male or female?

Maples: I don't know if that was ever researched, and I really have no feeling for how many women calls there were versus men calls. I have this feeling that there's always been a sense in the industry that women didn't participate as much as men with computers – with computer games, computer software, computers in general. I don't know if that's true or not. It's just that if you asked me what my intuition was, I would say that.

Management Approaches to Support

Ensmenger: As the industry is developing over this period, as the industry in general matures, it seems as if support is giving more attention to process, testing, and marketing – all of the business aspects of it. And it seems as if support is becoming a potentially larger and larger component of the cost of putting a product into the market. At the same time, it also seems there is a management problem with support, since support is way at the bottom of the hierarchy of employees at a high tech firm. Maybe documentation is a little lower, but support is low.

But you also have a competence problem. You want support people to be reasonably competent, but that makes them more expensive and it makes them more likely to get bored with that job. And so it seems that there are all kinds of very difficult management problems that might overlap with gender problems because you're talking about status. And there is certainly a cultural problem if you have a large number of support people who aren't quite real Microsoft employees.

Maples: I think a little bit of that is a business strategy. And I can give you examples of companies that do each of these. There is one thought that says, “I'm going to hire low-cost support people and I expect them to turn.” There is another thought that says, “I'm going to hire support people where I can siphon off the best ones into development." So I throw out the carrot that if you do well in support, your career path is into development. There is a third strategy that says. "Product support is a career in itself, and there are all kinds of tools and techniques and things to do to become professional." In fact, there are even college courses now in software testing, and you can make it a career and continually progress and get promoted in the management of the support group.

WordPerfect had an approach where they hired a bunch of students or wives of students; they were gone in 4 years. They could pay them relatively little. The students would go away after they were finished with school but there were a lot of them available. A number of companies tried to go out and hire people from schools that weren't top notch, or the students who were less qualified students, and make them product testers with the carrot that if they did well, they were able to advance. It's kind of like the farm team for the football team– we'll let you walk onto the squad, and if you make it, then you get to make the team.

Microsoft always looked at support as a career and tried to hire good people and tried to develop professional support people. Our belief was that the nature of a person who does support is somebody who is friendly and who likes solving problems. It's a person who might have been a nurse or might have been a good caregiver or something like that. And that's different from a developer who is focused on things, not people. The support person is a people person, a nurturer, and a developer is an introvert, and that doesn't change no matter how skilled you get or how experienced you are. So we wanted to deal with support as a career. There were support people who have moved into development, but it was pretty rare.

Abbate: We talked at lunch about the importance of the user groups providing sort of self help support for different types of software. Adam, it seems like you might have had more direct contact with this.

Green: Well, it depends on the kind of product. In the case of dBase, what I used to say is that dBase was a labor force. It was more than just a product and that became a very important sales factor. People would buy dBase because of the number of developers and consultants that were available – independents – and they would choose it even they felt another product was superior. So it became a very important aspect of the inertia of the product once it had this established base. The negative for the software developer is that the user groups became very, very vocal. A big reason for Ashton-Tate's demise is that when they delivered a very, very bad product as in the case of dBase IV, the user groups were there to say very loudly how bad the product was. You couldn't ignore them; the user groups were very important.

And the other thing the user groups did was help dBase become a generic language where there were many products that used the same language. So it transitioned from being an

Ashton-Tate product to being a language of which Ashton-Tate was one vendor. And there were five or six different products, each with its own segment of the user base. Nantucket had Clipper, which was more for the C programmer kind of style, the more professionally trained programmers. There was FoxPro, which had some appeal in that it was faster and didn't have as much of what some people would call marketing crap included. It was not as much of an end user product, but much more a developer's product. So the user base basically took the product away from Ashton-Tate.

Haigh: One thing I think that this is illustrating is the special characteristics of a program which will accumulate, if it's successful, hundreds of thousands of internal corporate employees and consultants, and build this huge community around it. And those are the kinds of products that accumulate many aftermarket tools and libraries and things like that. You wrote books on dBase, didn't you?

Green: Yes, I wrote the first book on dBase and a number of others.

Haigh: Was that aimed so that a naive user could pick it up to learn to install something using dBase? Or was it aimed at a more experienced user?

Green: The first book was very much aimed at a beginning programmer. Its title was the "dBase II User's Guide," but the real goal of it was an introduction to programming with dBase. What was unique at that time was that the majority of PC users had never used a computer, had never programmed. And there was a huge demand for programming; it was a very simplistic kind of programming, but it was still programming. And so there was a whole generation of self-taught programmers and a lot of career migration. That's ended now, but there was a whole generation of people who were managers, bank managers or middle managers in insurance companies, who started developing applications in dBase and decided that it was a lot more fun than their jobs. They would either come to my training courses or read my books or go to other people's training and then become an in-house consultant. Later they went out and became an outside consultant. Then there was a mini-recession in the late 1980s in IT, and they went back and became in-house consultants again. But it grew an entire generation around the product. All the training and the books were very much an introduction to programming.

By 1987 - 1988, client/server computing came in with network use of the database and that was primarily in the corporate market. With the corporate market, it became a retraining market where you started with COBOL programmers. The classic situation was a COBOL programmer whose company told him, "You are now a dBase programmer. Go and get some training, and learn how to become a dBase programmer." This whole group of self taught programmers was significant from 1981 through about 1986.

Haigh: Did Ashton-Tate or any of the later database vendors do anything systematically to encourage the growth of this community?

Green: It depends. You have to realize there were two teams who ran Ashton-Tate. Originally there was George Tate and his staff. George Tate was a good old boy. I don't even know if he was a high school graduate, but he was very friendly, very outgoing, a true salesman. He understood the idea of gathering people around him and he supported add-on products. He supported user groups. He would go to user groups a great deal. The transition came after George died in 1984. They went through a couple of leaders, and by 1987 or so, they brought in Ed Esber. Ed Esber was the classic MBA who didn't know anything about the product, had no respect for the product, and especially had no respect for the add-on market; he announced that he viewed the add-on market as parasites and felt that we were stealing their revenue. They actively drove the add-on market away, which is why, when they came out with a bad product in 1988, the add-on market in the community just said, "Okay, that's fine. We'll just go over to FoxPro and Clipper," and they were able to leave.

Documentation Writing as a Profession

Koblentz: I have a question for anybody who wants to answer it. Could somebody talk about the transition from documentation being just another task that you were stuck with to a distinct job?

Maples: Well, I think in the early days, the only documentation was written by the programmers. When they wrote the programs, they would also do some documentation. And maybe in 1984,1985, Microsoft went out and hired real professional people – Ph.D.'s in psychology and a number of others – and built real documentation. It kind of took on a life of its own. And then the documentation got very big; and led to computer based training and four or five different kinds of documentation; it got to be way overdone. So then it got scaled back quite a bit, probably in the mid-1990s.

Green: Yes. And that's where, again, the dBase market is unique in that. And I'm going to take some credit in that my first book on dBase was very successful. And we made the mistake of advertising a little too much, which told people how successful it was. So a lot of people became dBase authors and there was an entire industry of people writing dBase books so Ashton-Tate never had to worry about their documentation. They basically allowed this industry to support them, and that's continued among programming languages also. For example, I don't know how hard Microsoft would work on the C manual. You mainly just have a function library, and you allow other people to do the introduction to programming books.

Maples: And I think we've tried to move to that in most every situation by using the Microsoft Press and other people to write books that supported the products because there were a lot of different kinds of books needed. There's the beginner book. There's the expert book. There's the reference book. And we tried to do all of those, and some of them were task-based and some of them were function-based. Some people do better with a task-based book – you know, I want to create a letter to Mom. And some of the people are function based; they want to know how to do word wrap or whatever. Because of this, the outside book world can do it much more efficiently. One author with a decent royalty stream can make a good go of it, but it takes a department of ten people to do it in-house and it costs a lot.

Green: Spreadsheets are another case where the independent authors were very strong. Doug Cobb, who created Que books, was one of the first 1-2-3 beta testers. Because of that, he had an early start and wrote some very successful books. He then built an entire publishing company around it, and that model was then copied by others. The spreadsheet program was especially well-suited for this since with spreadsheets, people don't so much want to know how to do something, but it's more that they want to get a spreadsheet model for their particular application. This allowed the huge multiplication of documentation where people wrote spreadsheet models for engineering versus spreadsheets for accounting, and that leveraged the market.

User Groups

Maples: You asked the question about user groups. I think that you can probably find a good analogy to user groups in the automobile industry. In the 1910s and 1920s there were many automobile clubs and members all learned how to do things from each other. I think that's kind of what happened in the early 1980s in the PC world; the birds of a feather just didn't have anybody to talk to so user groups became important. By 1985, I think what had happened was that this influential end user had matured so that everyone knew somebody who was good at spreadsheets or DOS or whatever, either at work or in the neighborhood. So instead of going to a user group to learn how to do things, each person had a sphere of influential end users around him; these were the people who knew more and they were the kind of gurus who were willing to help others. And I think this is still in existence and that these gurus have a very big impact on what people buy.

Green: That's true. But coming up to the current time, the Internet basically killed the human user group; but at one point it was very strong. One other aspect of the user groups during the 1980s was they went through a transition from just being a computer user group to becoming more specialized. If you want to study user groups, Boston Computer Society at one point had 3,000 members, 5,000 members. It was very, very big. They originally were a single group, and then they split off by product because people became very identified with whether they were a TRS80 or Apple or CP/M users. And then they split off into particular application areas so there was a database user group, a word processing group, an accounting group and so on; people segmented very strongly. But there was a time when user groups were much more influential, especially out in the hinterland. If you went out to Houston or Cleveland or

Detroit, the user groups were very, very strong because company representatives didn't go there and they didn't have many computer stores. In those areas, the user group was much more of a networking group, not just social. It didn't just provide information, but it was a chance for consultants to sell their expertise or to network with other people. For example, doctors who were building a medical system would meet other doctors who were doing the same king of thing and they learned from each other.

Haigh: So how would you get the clients to your seminars? Would user groups play a role in that?

Green: In my case, I sold the book that I wrote directly to the users. I had a mail order company at the time, and we sold about 100,000 copies of the book. The actual history is first I worked in a computer retail chain, based in Charlestown in the Boston area; we had about 12 stores. I would do computer seminars in the back room because we wanted to sell Altos computers. We could sell a 10 megabyte computer – which was the top of the line in those days – for about $9,000, by giving them a dBase seminar. I would have ten people in, do a dBase seminar for them and sell two or three Altos computers. That was a very profitable business. I then turned the seminar notes into a book which I sold direct and I kept the mailing list from that. And once I had a big enough mailing list, I'd say, "Well, we've got 500 people in Detroit, let's run a seminar there," or in L.A. or in San Francisco. And so I leveraged that customer base in this way.

Abbate: So support and training sort of gets pushed out to the end user who then does it for free or has someone to do models in the office. Today you kind of expect to pick of the information in your office.

Green: That's right. One reason why the training market died off is that the average level of knowledge within any company was high enough so they didn't have to go outside for training. There were people in-house who knew the products.

Abbate: Do you know how early you got online support within the application where you could just hit a key to get information?

Green: Well, I think it was 1-2-3, where F1 was used for help. I think they invented that.

Maples: It was a pretty early standard. I remember in the 1981, 1982, 1983 time frame, all the design people said, "F1 is the help key" and that was included in all programs we did.

Abbate: Because it seems like you can now get almost 100 percent of documentation that way.

Ensmenger: In the VisiCalc session this morning, Bob Frankston said they had actually written help in VisiCalc and decided not to turn it on; but it was there in the code.

Printed Documentation

Haigh: It seems like, at least in the early 1980s, people expected that when they bought a piece of software, they would get one of those hard boxes inside of which was a ring bound book with pages inserted. And in the back there would a little folder with diskettes and there would be a number that they could call to get technical support. And they also knew that if they wanted to get a substantially improved version of the program, they'd have to buy an upgrade; they knew it wasn't a subscription. But if something was really terribly wrong with it, then they expected that they might be able to get a diskette with a patch on it.

Maples: I'm not sure the customers believed that as much as the vendors believed that. I think the vendors believed that if they were going to charge $500 for something, there better be something physical to deliver.

Green: Perceived value was a big issue.

Maples: And then, secondly, if they believed it had value, then they wouldn’t steal just the program because they wouldn't have the documentation to go with it. It turned out that most of the customers didn't care. They wanted the programs so they just copied them. They were inclined to do that but the documentation was an inhibitor. And some of the books were expensive. I can remember the first PowerPoint book cost something like $40 a copy; it was in five colors and there were all kinds of things in it that people thought they had to have. But now it has turned out that now all you get with the program is a little black and white pamphlet

Mahoney: It's an old joke in the customer support world – the cartoon of someone on the telephone saying, "I think they're going to open the book for the first time as they're calling in." [laughter]. The book just didn't get used.

Maples: At that point it was a bound book.

Green: Another aspect of the book industry at the time was the fact that the software was copy-able and was actively copied. And as an example, my mail order company would sell one copy of dBase and 20 copies of my book to a company. And then later they'd come back and buy another 50 copies of the book. So that's another big aspect of the book industry; we were in effect supporting piracy.

Haigh: That's an interesting topic. The question that I almost asked before was did the format that I described really begin with VisiCalc? Did VisiCalc have all those elements in their packaging? Did other companies come along and just matched them?

Green: If you talk to Dan Fylstra, he'll say that they were the first ones to come out with the softbound loose leaf binder. They had a standard package where they would have a brown loose leaf binder that was maybe 7 by 9 inches or something and that contained very good-looking documentation; I think they used glossy paper. You have to remember that at the time, other software came in a plastic bag; you would buy a cassette tape or floppy disk in a plastic bag with some Xeroxed pages of documentation.

Burt Grad: What's the timing on MicroPro with Word Star? When did it come out?

Maples: About 1978.

Grad: I thought they had that kind of packaging.

Green: Yes. They also had a soft loose leaf binder.

Maples: I think the first package that became kind of the industry standard through the 1980s was Lotus 1-2-3 in a standard-size 8 ½ by 11, slip covered box that was linen covered. They think they were the first company that really did a $500 product. Most of the products were in the couple of hundred dollar range before that and Lotus kind of set a benchmark. I think VisiCalc was $198 and most of the other products were around that price.

Green: Yes. And the VisiCalc people claimed that they were one of the first people to do a reference card that clearly explained all the commands. And one interesting design issue, which also is a documentation issue, is that they evolved the model of writing the documentation or the reference card first, and then insisting that the programmers match the documentation.

Maples: And they had keyboard templates. A lot of the products had keyboard templates so you would know what the function keys meant. WordPerfect did a lot of that.

Burton Grad: Did any of the people in the PC world look at any of the mainframe or mini products in terms of their packaging or what was done in terms of their customer support, or did they just create everything on their own?

Green: I know that some people had previous computer experience. For example, Dan Bricklin came from the mini computer market where he did a word processor. But by and large most people had come from audio retailing and from other areas, so they didn't have a mini or mainframe background.

Grad: One of the things we've talked about at times is this discontinuity between the mainframe, mini and the PC marketplaces.

Maples: But I think you've got to remember that there was a disdain for the other groups. The mini guys hated the mainframe guys. The micro guys thought the mini guys were just bozos.

Grad: Do you think it was conscious or did they think, "We're just starting and ours is so totally different, we've got to create our own?"

Maples: People thought, "We're smarter than those guys. Those guys are all trapped in tradition and everything else that's wrong, and we're going to fix that problem."

Green: Yes. The micro computer people thought they were smarter and the mini computer people thought that the PCs were toys. I taught seminars for DEC executives at the VP level when they decided to get into the micro market. And the most telling decision they made was that they gave it to the terminal division; this didn't make much sense since they would be obsoleting their own terminals with the new computer. They put it with the group that had no incentive to sell it, and they totally viewed it as a toy.

Maples: Well, I think also they came out and said that there's no place in the world for PCs; why would anybody ever buy one.

Haigh: I've seen from the very early days of the IBM PC that they put the official IBM PC service and reference guides in those boxes that were very similar to the size that 1-2-3 used. And I think the idea of writing the documentation first was being pushed in the 1970s in mainframe circles.

Selling Support Services

But there is another aspect of that comparison that occurs to me. It is clear that there were a number of people who were trying to do aftermarket consulting and training. I know that in some cases, at least in the enterprise software market, the product firms such as Oracle tried to make a bunch of money in services and training alongside the product. As I think about it, I can't think of any PC software firms that have made a big push in terms of doing the training and services that go with the product themselves. Is that correct?

Maples: The problem you get into is the price per product – how much can you charge for sales and support if you're selling a low-priced product. If you're selling products for $500, it's different than if you're selling them for $500,000. And if you buy an Oracle system today, you're going to budget several million for the product and several million for the support. So that's a good business. Nobody is going to budget that kind of money to support Word. As the micros get used more and more in businesses for large business solutions, I think we will find that there will be more consulting business.

The consulting services business is a very predictable business. Whenever there is something new and there are not many people who understand it, people are willing to pay a lot of money to gain knowledge. They gain knowledge by hiring consultants; they gain knowledge by going to school; they gain knowledge by buying books. As it becomes more and more mainstream, there's less and less of that. So if you track the consulting business, it follows the new innovations. It was the PC; it was client/server; it was the Internet and e-business. There have been many different places where there were spikes in the need for consulting and a lot of people and boutique consulting companies came about. For example, there were Novell network engineers who would break out and do consulting. But then, pretty soon it got to be not that hard to set up a little network, and they all went away.

Green: Or they went in-house; they went back to work for companies.

Maples: They go back and work for companies or wait for the next change in the technology where they can learn early and sell their services.

Corporate Help Desks

Mahoney: When did the corporate help desks come around?

Maples: The concept of the corporate help desk was formed with the need to reduce the price for volume sales. There was a little bit of concern in volume sales as there are some anti-trust laws about favored discounting, so we had to come up with some techniques to reduce our cost in order to pass the cost savings on. The idea was that we would transfer the first level of support to the buyer. If a company bought 1,000 copies and instructed all their people to call their own help desk instead of calling us, we could reduce the price we charged them because we saved support money. And since these costs aren’t accounted for very well, we could reduce the price as much as we wanted and felt we could justify that.

Mahoney: When did it happen?

Maples: It was in the early or mid-1980s, as soon as PCs started being prevalent in companies. If you look at the early PC penetration in companies, even through the first use of the IBM PC, they were almost all bootleg purchases back then. There were guys who would buy them on their expense account and bring them in the back door or pay for them in some funny way. In about 1983 or 1984, companies started creating policies on what PCs and what software employees should buy; with common software, employees could interchange documents with other people. And that's when the help desk concept came in. For example, they would say, “If you buy MultiMate we’ll support it; if you buy WordStar, we won't support it, and you're on your own.” There was a period of time in about 1985 or 1986 where the IT guys regained control of the PC phenomenon and then they lost it again in the late 1980s. I'm not sure why, but it kind of went back to the users and away from the IT shops. As a result, the IT guys started getting forced into supporting whatever the user thought was best.

Green: That's one of the reasons for the rise in client/server systems. The IT guys realized that there was a huge amount of data that they didn't control. There was the budget, for one thing, and there was also the data. All this data existed at the departmental level and there was no way to control it. So the idea of running a wire down to the department became very important because then the IT group controlled the data. And once they controlled the data, they had to control the hardware that the data is residing on so they controlled the budgets for that.

Maples: I think it was also the result of the need to insure data accuracy. When everybody had their own spreadsheets, they could add data about anything and make it look halfway official. By centralizing the data, the company was able to have more consistent data for making decisions.

<u>Computer Based Training and Instructional Wizards</u>

Grad: Did you talk about computer based training at all? That was one of the areas in which the computer was supposed to make a big difference for any kind of major training need. We could use the word "wizards" now as the current name, but IBM was writing computer based training programs in the mid-1970s and 1980s, and a number of companies were also producing courses. I was wondering whether anyone here was involved in that.

Green: From my experience they weren't successful, and I think a major problem is the generational problem in that at least my generation did not want to read on the screen. Now the Internet has changed that to some extent, and certainly a younger generation is more interested in reading volumes of material on the screen. However, earlier, it was the case that people would not read a lot of text on the screen; they'd much rather read a book.

Grad: Computer based training isn’t about just reading; it’s about a way of practicing doing something.

Maples: We spent a huge amount of money on computer based training and it was relatively ineffective. It's hard to figure out. This gets back into task-based versus function-based instruction. It is the difference between are you trying to learn how to do mail merge or are you trying to learn how to use some features. So we couldn't even figure out the best way to do it ourselves. And at every release we would change the whole strategy and redo everything again. Pretty soon the training in some companies was two-thirds computer based training, but the majority of users never used it.

The majority of users want to be productive relatively quickly and they would like to be able to learn how to do things in small increments. If you study how people learn computers now, it takes a while to get comfortable with about 10 to 15 features. It doesn't matter if they're products or features or what. But as soon as they get comfortable with between 15 and 30 features, their learning becomes almost synergistic. They try things and learn things as they go. So the trick is getting people over those first few things and that's the way people want to learn. So computer based training where you sit down and spend two days learning how to do everything was never used by the user very much. In addition, it was very expensive to create.

Grad: There were some proposals to do on-demand training, using the approach of computer based training as a method of teaching. You'd try it, you work at a problem and it tells you if you're doing it wrong. It shows you what to do to correct it. The idea at one point was to try and do that on demand. If you wanted to learn how to do a task, mail merge for example, you could right then go into specific interactive training for that procedure and you would practice it.

Maples: The help systems have gotten pretty good at that – the wizards and the help systems.

Mahoney: I was going to ask you about the wizards. Has there been research about how much they're used; how many users turn off that damned paperclip?

Maples: That's gone now. It was more in the way than it was a benefit, and it doesn't come up now with the new products unless you tell it to. The wizards were very short – for example, "Here's how you make a business card. Do these steps and use this template, and it will do your business card." If you wanted to do a fancier business card, then you had to learn a little bit more. The wizards were first done in Publisher; they were named by the product manager and publisher. That was the first product, and the name just kind of stuck. All of the Microsoft products picked up the name wizard, and it was kind of a toy name at the time. And the business guys thought it wouldn't work very well with the business applications but it worked fine.

Grad: Isn't that an on-demand kind of computer based training in some sense of the word?

Maples: Well, a wizard is computer based training for a small task. It's not computer based training for a product. So if you want to learn how to do mail merge, there is a wizard that takes you through mail merge. If you want to learn to do word processing, there isn't a computer based training program to learn how to use Word now.

Green: It was training as a more multi-level interface, in that it gave you an easier interface than knowing the menus for certain tasks.

Grad: You make an interesting point: it actually carries you through the process. The question is do you learn the process to do it on your own in the future, or do you simply go back to the wizard and get the information the next time you need it?

Maples: No, I think it's always a question of how many times you use it. I think you use the wizard the first two or three times, and if you use it every day you won't use it any more after that. The thing in the help system that they've tried to spend a lot of money and time on is how to take a question that is stated in human terms and create the way to get into the database of information that's there – in other words, the linguistics of dealing with a question so that you know what question they are really asking. For example, if they ask which printers are supported by da, da, da, da, you need to know which are the key words and where to point in the database to get the answer for them. There's a lot of research that has gone into that, and I don't know that I'd say it's 100 percent successful, but it's an awful lot better than it was. It's still got a long ways to go, but I think that if you use the wizards in some of the Microsoft products to ask questions, you'll get pretty good answers.

Green: One other thing to throw into the conversation is that documentation and help systems and support are all assuming that products are equal. But what you really have to remember is that the learning curve is different for each unique kind of product. I don't mean the brand, I mean the application type. They're distinctly different, and that affects the kind of documentation and training you need.

There are three major types of applications. In word processing, there's a minimal learning curve where basically you learn how to type and you learn formatting commands. That will solve 99 percent of what you need, and you can just stay at that level. And that's why we said mail merge, for example, is when they reach this peak. All of a sudden they have to do a mail merge. They've never looked at the docs, they've never needed to, and now they hit a total wall. But in most cases what they'll do is they'll bounce off that and say, "Okay. Well, I just won't mail merge. I won't go there."

Maples: Yes. Or they want to do tables or they want to do a table of contents or an index, or they want to have chapters or they want to have numbering of paragraphs or something like that.

Green: So in a word processor you learn a minimal set of tasks and then have an occasional discontinuity where you say, "I have to get over this little bump in the learning curve." And I think that's one reason why you don't get as many books on word processing because the basic tasks are so easy no one is going to buy a book for the occasionally hard task.

Spreadsheets, on the other hand have a gradual slope where it's very simple at first to set up a basic spreadsheet. Then they build a bigger model and a bigger model and a bigger model, and all of a sudden it starts getting more complex. They need macros and they have to start transitioning into programming. That's when they can use a book to help learn how to make that gradual transition.

Databases are totally different, especially the program databases, because there's a very abrupt wall when they first start using the product. They can't do anything with the product until they learn a great deal. Books are most popular with this type of application because users realize that they have to study first before they even open the box or else they won't get anything out of it.

Bug Fixes

Haigh: I have a question about the distribution of emergency bug fixes before the Internet. I know WordPerfect had a reputation for good customer support. If there was something that didn't work, you'd see the company at the trade show and they would give you a diskette right there that would fix the problem. But clearly, in the days before critical updates and online things, we didn't have a mechanism between major releases to get those kinds of bug fixes out to people. And those releases only happened every two or three years. So how did that evolve? How were mistakes that were made in products dealt with?

Maples: In the early days, they simply said, "Wait for the next release; we'll fix it in the next release." As you got to servicing more commercial or large enterprise customers, you would do it through their help desk. You would send them a list of problems and a list of fixes, and it ended up being distributed to whoever had it.

The thought was that distributing fixes was more of a hassle than a benefit because most people didn't run into problems. A high percent of the real problems had to do with a particular hardware configuration. You know, if you had this display adapter and this display and this printer, there was a conflict in IR assignment or something. So if you sent out a fix to that problem, you sent it to 100,000 users but the problem only affected something like 2,000 users. So fixes were generally just not sent out.

Now application fixes are sent out every six months or so in bundles. And in the meantime, if you call with a problem that's a real problem and you talk to somebody, they'll tell you what to do to work around it or give you a fix. But by and large, you won't get a fix on the Internet or anything else unless you find that problem yourself. But the developers of the higher end stuff like the operating systems are always sending out fixes one at a time. If you find a bug in Internet Explorer that a buffer overruns for example, there is a patch for it. And they may send you five or ten a day when somebody finds something. This is done through automatic updates.

But if you look at the applications, most of the bug fix releases come out about every six months. However, if there is a special push, for example if somebody was raising Cain about the security defaults, they'll send out a patch to change everybody's security defaults, just because people complained about it.

Green: In the smaller companies, especially in the early 1980s, it was more an on-demand thing. As someone would call in with a problem, they would send out a disk to fix it. There would be regular upgrades for which they would either charge or, depending on how long you've had the product, they would send it out for free. But if you called Peachtree Accounting or MicroPro for WordStar, they would send out a disk. At Ashton-Tate, initially George Tate would go and get Wayne Ratliff, the author, to fix the bug and he would mail you a disk personally. He built his customer base person by person in the first year or two. And they did that even when they were selling several thousand copies a month.

Maples: I think there is an interesting side question here, and that's who puts out the release – the fixes. I think in the early days, the developer who created the program was the guy who fixed it. Then companies went through a phase where they would have a development team and a fix team; they had one set of people who just worked on fixing problems and another set who wrote the new releases and the new code. At least in Microsoft now, the bug fixing goes back to the guy who wrote the code the first time. So if you wrote some routine or something that has a bug in it, then you had to fix it. And even if you're working on a new product, you have to go back and fix the bug. The purpose of that is to create the discipline of trying to do a better job in testing and releasing to start with. If you write the most bugs, you have to spend the most time fixing the bugs. But I think that an alternate philosophy says, "Let's hire some programmers who are junior and not quite as good and have them fix things. And we'll take our very best programmers and have them work on new stuff." With this approach, the fixes usually don't get into the new stuff and you just repeat the same problems, and the developers don't learn.

In-House Testing

Abbate: How much in-house testing was there and how long did it take for it to become a standard to have a whole group of people testing products as opposed to developing them?

Maples: I think it happened early, but the numbers changed dramatically. If you take a graphical end user application, in the early 1980s there may have been one tester for five developers. In the mid-1980s it was one to one. At the end of the 1980s it was two testers per developer. This made us start thinking about the cost of testing. You have two testers per developer, and every developer is spending half their time or more debugging things. So they spend a little bit of time writing specs, a little bit of time writing code, and they spend a lot of time integrating and testing. So of your total organization, you have two and a half out of three people working on bugs or working on problems. So it's important from a process point of view to figure out how to fix those bugs earlier and have developers not write bugs. It's a lot cheaper not to write bugs than it is to fix bugs. That goes all the way back to the spec and the design documents and a lot of other things. I think that if you went to the well-run programming shops today, you would see a lot more emphasis on writing clean code than trying to figure out how to add more testers to do more testing.

Koblentz: What are today's or tomorrow's documentation writers going to learn that you guys did the best?

Maples: I think the major thing that we learned is you can spend a lot of money and not get much benefit. And I think that there will always be tech manual people, and they'll write manuals. But I think as a percentage it will stay relatively small. We got where the number of developers was smaller than the documentation groups. I think now it's probably one documentation person for five or ten developers. And it would take us 20 documentation people to write a good manual to ship with product, but you could get 20 people on the outside and they'd write 20 books.

Green: And the outside people would write different kinds of books. One of the problems for documentation writers is that they have to write a reference guide, typically alphabetized by command. And you can only then write a small users guide, which is more of a getting started manual. However, someone writing an outside book can assume that there already is a reference manual so they can be much more concerned about actually teaching somebody how to use a product. This means they can construct some kind of logical learning curve where they don't have forward references all the time, where they can use a consistent example that builds over time and where they can be application-specific in some cases. So I think the problem is that the constraints of in-house documentation writers don't allow them to get any better. They are locked into what they have to do.

Maples: Unless you want to try to do it all. Some people would like to try. There are a lot of people who would like to have one book that is task oriented, another that is function oriented, along with computer based learning and a bunch of reference cards and keyboard templates.

Green: But this leads to diminishing returns. As we said, the users don't read the documentation. Except for a programming language where the publisher has to publish a function manual since no one else knows the detail, in most cases documentation is really more for show. The customer expects it so you do it, but you know they're not going to read it anyway.

Outsourcing Documentation and Support

Maples: I think there may be a trend to outsourcing documentation. I'm not 100% sure of this and it would be worth doing research on it, but I think there are now a lot more independent people who write the documentation than in 1985 to 1990.

Koblentz: Is that one of the reasons why some documentation is so bad: the people who write it are different than the people who originally created the program?

Maples: Probably, but it may be bad because they just don't care. And it sort of goes back to the fact that often it's translated – we get a translation of a Japanese document and it probably doesn't read very well. And I think some people just don't care.

Haigh: How about outsourcing technical support centers, other than effectively outsourcing to an end user's own corporate help desk?

Maples: Well, what Microsoft did, starting before 1985, is that we'd outsource the support center as the products matured. So in 1986, 1987, we were working on more Windows stuff so all the DOS support got outsourced. And then the games were outsourced. I think it depends on the complexity of the product and the complexity of the buyer and how hard the questions are. If you think it's going to be the consumers who call in, it's easier to outsource support than if you think it's going to be the IT professional with problems that will turn into a bug.

I think there's a weird set of debates going on right now about foreign outsourcing, about sending stuff to India and things like that. Everybody likes to debate it because they don't want to lose their job, but nobody wants to pay higher prices for their products, either. And I think that everybody is going to recognize that if a job can be done somewhere cheaper, it ought to be done somewhere cheaper. We've been outsourcing jobs from America for 50 or 100 years. How many textile plants are left? How many shoe manufacturers are left? That's not a new phenomenon. And hopefully you keep upgrading your skill levels to do the things that you need to do, and then outsource the things that are less important.

Green: I have a general issue on support, and this is not meant as a knock on Microsoft. Microsoft has different needs because of their scale. As a result, you can basically study Microsoft and then you can study the rest of the software industry and that's just the reality.

Because of the scale of Microsoft, they are forced to view support as either a cost or potentially a profit center. In smaller software companies, the scale is somewhat smaller and support is viewed as a sales task that you could use to sell the customer moving up to a more powerful product or a new version of the product. And I know even Microsoft people are trained to up sell. But there's much more sense that support is part of the rest of the team in smaller companies.

The other thing that I think is very strong in small companies is the idea that support is where you get your design ideas. Your users are telling you what they want at the same time as they are complaining. So they're not just saying, "Gee, this doesn't work." They're saying, "Gee, I wish it could," and that's very valuable information. And so in smaller companies, support people either are, while not the primary developer, members of the development team or members of the sales team who really try to learn from what their customers are telling them. Did you have any feedback mechanism in Microsoft to get this type of information?

Maples: I think it's a barbell. I think the very small companies get good feedback. And then, as you get a little bigger, you get separated. You have organizations that do support and others that do development. And then, as you get really big, you implement mechanical aids; you automate the feedback process. Microsoft does an excellent job of getting feedback from the product support guys back to development; there are systems in place. This has been going on for 10 or 15 years.

Koblentz: Another interesting trend is that with some of the Linux companies, because the product itself is free, their entire revenue stream is from support and tools and things that go around the product. But the supporting manuals are the number one feature of some of these companies like Red Hat, for example.

Maples: Originally the whole PC industry was geared around trying to reduce the cost of support because they were making money by selling or licensing intellectual property. Then what developed was what's called the free movement which considered the intellectual property to be free and the support and distribution is what you charge for. The software companies thought this was the dregs part of the business and they wanted to get out of the packaging business and make it as cheap as possible or make it available on the Internet. And they wanted to get out of the support business by having support done through automated databases.

Haigh: You mentioned the goal of cutting the support costs in half with each release. Was that broadly accomplished?

Maples: Yes. We used to do quite a bit better than that. And that's not cutting it in half. That's cutting the copies by about a factor of ten, and support by a factor of five. But it's a scale thing that as you increase the scale, you're always reducing the cost as a percentage.

Green: In the mainframe industry, support was a big business and you put the SE in for support. With PC, we actually tried to release products that needed a minimum of support. There's the classic story about VisiCalc that if they couldn't explain a feature on the reference card, they would change the program.

Welke: But there was a point where the mental set was contra of that. I can remember Pete Peterson taking me in through his cavernous room in Utah, showing me 600 Mormons answering the telephone [for WordPerfect].

Green: Mormons knew all sorts of languages, so they could do international support and that was a real strategic advantage for WordPerfect. It wasn't an accident, and they wouldn't have to pay as much for them.

Welke: I wonder how many software development companies heard Peterson and saw the WordPerfect example and said, "It looks like a good model. We ought to follow it."

Green: I think the general conclusion was, "We've got to make sure we don't need so many people in support."

Maples: Well, they did two things. They had a lot of people and good people, but they also had a toll free telephone number for support. When you have long wait times tying up a lot of toll free trunks, it gets very expensive. Microsoft never had free telephones; customers always had to pay for their own calls. So we focused on getting the call answered more quickly because if a guy was paying for the phone call, it was important not to keep him on the phone call for 20 or 30 minutes. And to some extent it's just better for customer satisfaction. With WordPerfect, the call was free and they had longer call times. But WordPerfect did a great job of marketing that. It was one of their key marketing points.

Green: But how often was support passed on to the manufacturers and the distributors and others?

Maples: In all of early days, the PC guys believed they could get the re-sellers to do the support. It was a total disaster. The re-sellers turned over too quickly. They would do whatever they could to get the certification. They were selling the software and they had very low margins. And you've seen the re-seller margins get down to 3 or 5 percent now, and it's because they don't do anything. They don't provide any real service; they just kind of store the product and demonstrate it, and that's only worth about 3 to 5 percent.

Welke: We're done. Thanks, everybody.