PC Software Workshop: Lotus 1-2-3

Moderator:
Jeff Tarter

Recorded: May 6, 2004
Needham, Massachusetts

CHM Reference number: X4310.2008

Table of Contents

PC Software: The First Decade – Lotus 1-2-3 Workshop

Conducted by Software History Center—Oral History Project

Abstract: Jonathan Sachs, the primary developer of Lotus 1-2-3, is joined by three other former Lotus Development employees to talk about what made Lotus 1-2-3 such a significant product. Starting with the selection of the IBM PC as the platform, they talk about the development process and the features that made it a top-selling product. They discuss the management changes and the problems resulting from the company's dramatic growth that led to Lotus 1-2-3 losing its position as the number one spreadsheet. Finally, they discuss the competitive products and what led to the problems and ultimate sale of the Lotus Development Corporation.

Participants:

Name	Current Affiliation
Michael Kolowich	DigiNovations, Inc.
Chris Morgan	C. Morgan Comms.
Jonathan Sachs	Digital Light & Color
Scott Tucker	KMT Software, Inc.
Martin Campbell-Kelly	Warwich University, U.K., historian\
Thomas Haigh	The Haigh Group, historian

Jeff Tarter: Tom [Haigh] and Martin [Campbell-Kelly], would you start by explaining your interest in the Lotus Development Corporation as historians? Then the rest of you can briefly introduce yourselves, please.

Background

Tom Haigh: I'm one of the two historians assigned to this session. I've worked with the Software History Center on its ADAPSO history project. I'm currently a consulting historian. I have a doctoral degree in history and sociology of science from the University of Pennsylvania.

Martin Campbell-Kelly: I'm Martin Campbell-Kelly. I teach at the University of Warwick. My interest in this topic is that I'm planning to write a book on the history of the spreadsheet, this all-American story.

Jonathan Sachs: I was one of the early employees at Lotus, and I wrote pretty much all of the original version of Lotus 1-2-3.

Scott Tucker: I'm currently with KMT Software, which continues to benefit from the legacy of Lotus 1-2-3. I joined a tiny company in Connecticut named DSS Development, later named Optionware, which was one of the very first companies in the Lotus 1-2-3 aftermarket. I joined that company in 1983, and I remember beta testing Release 1a of Lotus 1-2-3. Because it was my very first job, I just plunged headfirst into it and probably spent 14 hours a day for a year or more completely immersing myself in Lotus 1-2-3 in every facet, especially because we were trying to stretch it to its limits. Within a month and a half, I was training our clients in how to use spreadsheets, and within three months I was leading a development team—my very first job out of school—to create 50 business applications which we intended to release to the market. We were led at that time by this crazy English guy who thought that templates and business applications were going to be the next big thing. Shortly after that, in 1985, I joined Lotus, and about nine months later I joined the actual design team on 1-2-3 Release 3, so I came rather late in the history of Lotus.

Michael Kolowich: I'm also relatively late in the history. My first encounter with Lotus 1-2-3 was as a user. I was a consultant, then a manager, at Bain and Company, a strategic management consulting firm, and basically led the company in adopting first VisiCalc and then 1-2-3. Later, I became a partner at the firm and became a consultant to Lotus, particularly concerned with the early thinking on electronic distribution but also on other issues. In early 1985, I joined the company, although technically I was on loan. I went on a leave of absence from Bain to become corporate vice president of marketing and business development for Lotus, and held that position for about a year and a half. Then, the company reorganized, as it periodically did, into product lines as opposed to functions. When Lotus became a product line organization, I had responsibility for everything in the company that wasn't a spreadsheet: graphics, text, and information management products.

Chris Morgan: I came on board at Lotus in January 1983 as vice president of communications, but I had known Mitch Kapor for several years prior to that, and I was editor in chief of *Byte* magazine in the early1980s. In 1981, I got a letter from Mitch telling me about this new company he had formed. I was covering Lotus, indirectly, since I was covering the spreadsheet industry; and I knew Bob Frankston and Dan Bricklin in the early1980s. I used to hang out with all these crazy people in Cambridge, and Lotus made me a job offer. I came to Lotus in charge of documentation, public relations, and marketing communications. I was there from 1983 to 1987, although about two years into it I switched over and started Lotus Publishing and we started *Lotus* magazine.

The IBM PC Platform

Campbell-Kelly: To what extent was it obvious in 1982 that the IBM PC and the IBM compatible PCs were going to become the dominant platform?

Sachs: In the early days of Lotus we were in a basement at 180 Franklin Street in a rather seedy part of Cambridge. One day, Mitch somehow obtained one of the very early IBM PCs. We all gathered around, and there was a pretty clear consensus that "this was it." We were in a kind of holding pattern, doing some preliminary spreadsheet prototyping and waiting for the right platform – it's like surfing and waiting for the right wave. The fact that it was an IBM PC—IBM sort of legitimized the personal computer industry. Data General had some sort of ad about a company that had announced they had entered the business and now they had "legitimized" the minicomputer business. The tag line was, "The bastards say, Welcome." The fact that it was IBM was critical, and it was clearly a step up from the Apple in terms of technical capabilities, the ability to add extra memory. The display was very high quality for text—way ahead of its time. So, it was pretty much a no-brainer at that point. It wasn't obvious that it would be the dominant platform, but there were strong indications.

Campbell-Kelly: Someone said there were versions of 1-2-3 developed for other platforms. I seem to recall seeing one for the DEC Rainbow, for example.

Tarter: Which was an MS/DOS machine, right?

Sachs: Yes, because there were no versions other than DOS until much, much later.

Haigh: As I understand it, one of the distinctive features of 1-2-3 over some other new spreadsheets being introduced around the same time was that Lotus targeted the IBM PC hardware at a low level instead of trying to work at a generic DOS level.

Tarter: I was going to ask the same question. You decided to go around the video, so you'd be 100 percent PC compatible instead of a generic PC running Microsoft DOS?

Sachs: Yes. I was very concerned about performance and efficiency, and there was a BIOS built into that computer that certainly encapsulated some of the low-level functions. Spreadsheets are extremely screen intensive; they're always updating the screen, and the ability to update the screen really quickly pretty much made it necessary to go behind the BIOS routines and address the hardware directly.

There were other cases where we used the BIOS routines, for accessing the keyboard and other things that weren't as performance critical. The screen is probably the only thing that we went straight to the hardware for.

Haigh: But, that's still the question, as it's explained in the book *Accidental Empires*, for example—at that point Microsoft believed that there would be a market for DOS computers that weren't 100 percent IBM PC compatible. I believe the DEC Rainbow was one of those—there were some others—and Microsoft's own application suite required only DOS-level compatibility, not even BIOS-level compatibility.

Sachs: All of that was irrelevant because the IBM PC had 90-some percent of market share and nobody cared whether you supported the other spreadsheets or not. In fact, the very first thing we did after Version 1 was to make Version 1a which had drivers so it could be customized to run on the whole crop of other "somewhat IBM" compatibles that came out: Texas Instruments, Hyperion, and Victor—all kinds of companies that have since gone out of the PC business. Mitch felt that it was important to support a lot of these so that it didn't create a market opportunity for somebody else—if we didn't do it, then they would do it. The drivers provided a way to encapsulate the platform-specific code.

Lotus 1-2-3 Performance

Morgan: One thing that I did notice in going back and looking at a lot of the early reviews of 1-2-3 was the comments on fast performance. That seemed to be a very noticeable advantage for a lot of the reviewers.

Kolowich: This became an issue later as 1-2-3 was revised, particularly with Release 3. When it was re-implemented in C, a lot of that performance was lost. The performance degradation that came from working at a different level caused the software to kind of get rejected by beta testers and others, so that it kept needing to be re-engineered and rebuilt. I don't think it came out for three years.

Tucker: Yes, it was originally supposed to be an 18-month effort, and it stretched out a long time so we could make it fast. And, yes, as much as writing directly to video might have been important, it was the minimum recalculation that Jon implemented for the first time in a spreadsheet, as far as I know, that also had a great deal to do with this.

Sachs: Actually, there were other implementations of the so-called natural order of recalculation. I knew that people had done it; I honestly had no clue how to implement that. Eventually we hired a fellow named Rick Ross who had done his PhD thesis on garbage collection in LISP. There was a one-to-one correspondence between that problem and the minimal order-of-recalculation problem.

Tucker: I'm sorry for confusing this by using the word "minimal." You were right. The natural order of recalculation is what was going on there.

Sachs: Rick Ross worked this out. He kind of recast his algorithm in a form that I could use, then I implemented that within 1-2-3, and it worked very nicely. The other thing he did was to create a customized version of 1-2-3, which took its input from a tutorial script (which came with 1-2-3) combined with the keyboard, instead of taking its input solely from the keyboard. Other people wrote the tutorial scripts that ran with it. Later, he went on to be a developer of a product called Reflex, which was popular for a while but never quite caught on. I lost track of what happened to him after that.

The Development Process and Tools

Campbell-Kelly: How did those production methods evolve over time? When you started, it was just you writing the machine code. There was a phase where you switched to C. I'm presuming there were a lot more developers at this stage. Can you say, over an 8- or 10-year period, how the development process or tools changed and how the process became bureaucratized?

Sachs: I would have preferred to write in C right from the beginning. There were a couple of early C compilers for the IBM PC, and I ran some benchmark tests where I hand coded some routines and let the C compiler compile them, and there was a factor-of-five difference in size and presumably speed. So, I said I guess they're not ready for prime time. I had a lot of experience writing in assembly language prior to this project, so for me it wasn't as intimidating as it might be for most people.

Campbell-Kelly: But, you did actually use a PC as a development system although I think some people early on had used minis and emulations.

Sachs: Yes, unlike VisiCalc, which was all done by cross-compiling.

Campbell-Kelly: Why was that?

Sachs: The tools were there. IBM released a full suite of tools, and they released an assembler and a debugger. I wrote my own text editor that I converted. And, there was enough *there*. Even with just a floppy-based machine, it was perfectly usable.

Campbell-Kelly: Even without a hard drive.

Sachs: Yes. I got the first hard drive when they came into the company. It was about the size of a toaster, and it was a 5-Mbyte Corvis hard drive, a big, white box. When you're working with floppies, 5 megabytes looks very sweet.

Haigh: How big would the development team have been in the very early days?

Sachs: Just one. Well, that's not completely true. When I joined Lotus, there were two other programmers, one of whom was 12, and one was 13. They were doing development for the Apple II. But as far as working on 1-2-3 was concerned, it was just me until pretty late in the game.

Tucker: I understood that you had implemented 1-2-3 in C, and then for performance reasons decided to rewrite it in assembler?

Sachs: What I did when I first joined Lotus was to do this prototype, and that was actually written using a C compiler running under CP/M on an Apple. That was probably the fourth spreadsheet I had already written by then because my first exposure was when I was working at Data General in 1976 or 1977. I wasn't there, but I heard about a meeting where Dan Bricklin and Bob Frankston came by and demonstrated VisiCalc to our department, trying to get them interested in a Data General version of VisiCalc. One of the managers who came out of the meeting said, "Oh, it's really trivial. You know, there's nothing interesting in this; we can do something better than that ourselves." And, of course, they never did anything. After I left Data General I was contracting, and I was approached by a guy who was still at Data General but planning to leave named Alan Clutchman. He was interested in paying us to write a VisiCalc clone that ran on Data General computers, and that was my first exposure to spreadsheets. I wrote that in Data General assembly language, and that product was sold for a while. My partner at the time was John Henderson. He and Alan never got along particularly well, to put it mildly. Alan wanted us to do a version for DEC computers. But we never saw significant revenue from the Data General version, and it didn't look like we'd see any from the DEC version, so we just told him to go away, and he hired somebody else to do it.

Actually, we were interested in pursuing this a little bit, and I spent some time rewriting the spreadsheet again in another language called BCL, which is a kind of obscure predecessor of C, in preparation for maybe porting it to the Intel 8080. After that version had been created, Henderson and I started to get on shaky ground because the company was built on contracting revenue, and what I was doing wasn't producing any revenue, and what he was doing was. So, that really wasn't fair to him. We decided to go our separate ways, and we basically shared that code. I got to take a copy of this version that was rewritten in BCL with me and to have rights to develop successors to that.

Tarter: Was that code basically your equity contribution to Lotus?

Sachs: I brought that to Lotus with me, yes. The connection with Lotus was that Henderson somewhere along the line had met Mitch, and Mitch knew that we were working on this spreadsheet, and he came by and talked to us. Henderson was pretty unimpressed with Mitch. Henderson saw things totally from a technical standpoint, and Mitch was a little flaky on the technical side. Anyway, after we split up, I had a list of things that I might pursue to leverage what I had already done. I met with Mitch in Cambridge. What Mitch and I shared was this

excitement about spreadsheets—thinking it was a really big thing and wanting to make a better mousetrap. That shared energy is what got us going.

Tarter: Did you approach Mitch as opposed to him coming and approaching you?

Sachs: Yes, I approached Mitch, other than the fact that he had approached us as a company prior to that.

Lotus 1-2-3 Features

Haigh: It's been said that the feature list for 1-2-3 in large part was the feature list for the advanced version of VisiCalc, which was never really produced properly. Do you believe that that's true?

Sachs: We were vaguely aware that the advanced version of VisiCalc was being developed. I'm not totally sure of the chronology—both Mitch and I felt "Gee, I can do it better myself." So, I don't think that the 1-2-3 features were the features of the advanced VisiCalc. Neither of us was averse to stealing a good idea, and certainly things like the database part of 1-2-3 didn't occur to either of us until Context MBA came out. But, the connection with the advanced version of VisiCalc was pretty tenuous.

Tucker: Maybe another way to ask that question is where did the feature set come from? Was it driven by, "This is all we can implement in the time?" Or, "This is all I know how to implement, and therefore we can't do this and this?" Because that's often the way engineering goes.

Sachs: Before Lotus—and before it was called Lotus it was called Micro Finance Systems—Mitch had worked for Personal Software, which was the publisher for VisiCalc. He had written—himself or with the help of some other programmers—two programs that worked with VisiCalc, one was called VisiPlot and the other was called VisiTrend. They extended (or worked with) VisiCalc to add graphing and trend analysis. So the idea of integrating graphing with a spreadsheet was a natural extension. Right from the beginning, that was the 2 in 1-2-3.

The 3 was going to be word processing, and I spent several weeks taking a crack at that in the middle of the development process. And, it wasn't going very well. Right about that time, Context MBA came out, which had a database function. I looked at that and said, "Gee, I can do one of those really easily." It took just a few weeks to throw that in. Also, it wasn't just that it could be done easily, but that it clearly had more synergy with the spreadsheet than word processing. So, all that was left of the word processor in 1-2-3 was the range justify command.

Campbell-Kelly: Do you have a sense of what proportion of 1-2-3 users actually used the

database feature?

Tucker: I think relatively few used database in the sense of hard core querying and so on, but lots of people keep lists. A few Excel features were added to Lotus. Microsoft had researched the market and tried to understand how many people used these features, and the auto sorting and pivot table and so on are Excel's ways of using database-type functionality in a spreadsheet; but very few people use a Lotus spreadsheet as an actual database, as a form for entering and capturing data. But, being able to do some analysis, aggregation, sorting, and so on—people use that all the time.

Morgan: To add to what Scott says, he's absolutely right. Readers of *Lotus* magazine tended to be really deeply into the products, and they would use a lot of the features. As I recall, most people using 1-2-3 in the early days were mostly using the spreadsheet, period. Sometimes they'd use the graphics, and sometimes they'd use the database. But, if you asked them to describe the product, they would call it a spreadsheet; they wouldn't call it an integrated package of three different things. That's just because that's the way people tended to think. They still do it today; people use maybe 2 percent of what's in Microsoft Word.

Tucker: This touches on a very key point—a conceptual point within Lotus as to what people use the product for. Is 1-2-3 the seed of a product that is blossoming, or is it the product? It was looked on as integrated software. People did look at the 1, 2, and 3, and take that seriously. In many ways, I think Lotus at several times over its history believed its own PR too much, and this was an example. As a result, Symphony grew out of it. When I first joined Lotus, I was a bit of a history buff myself, trying to understand what had gone before it. I was able to locate very early versions of Symphony back when it was called—internally—1-2-3 Release 2 and used a word processor in the spreadsheet, but it still looked very much like 1-2-3. It was fascinating, looking at this transition that was happening, but that's not what the market was doing. That's not what users were doing.

Campbell-Kelly: Can you tell us a bit about the evolution of Lotus Release 2? Going back to that previous question about how you developed this further. What was the structure of the development process, and how was user opinion integrated into this process?

Evolution of the Product

Tucker: We're skipping, of course, Release 1a and Release 1a*.

New Speaker: Right. And the other 15 slipstream releases, right?

Sachs: I think Symphony came before Release 2. Let me try to remember the chronology. Version 1 was the first release.

Tucker: We were showing it at the Fall 1982 Comdex.

Sachs: Yes, we were showing Version 1a, which added support for drivers and for running 1-2-3 on different platforms shortly thereafter; then there was kind of a pause and we were working on Symphony, on the theory that if three functions were good, five were better.

Campbell-Kelly: So the Lotus development team stopped doing 1-2-3 and started doing Symphony?

Disbanding the 1-2-3 Development Team

Sachs: 1-2-3 was totally abandoned, and all the work went into Symphony. The expectation was that it would completely replace 1-2-3, that no one would want 1-2-3 after they saw Symphony.

Tucker: I think that was the decision that basically set Lotus's history—the decision to disband the 1-2-3 team. That process seemed to happen over and over again at the company, and I'm speaking now out of frustration. It was just such a sad thing to see—that they disbanded that team and then went to Symphony as Jonathan was describing and didn't come back around to 1-2-3 Release 2 until after they realized that Symphony was not going to be the successor product to 1-2-3, but in fact had to be a parallel path.

Kolowich: The company was religiously tracking penetration numbers and upgrade numbers. The expectation was that there would be a near 100 percent upgrade from 1-2-3 to Symphony. I don't believe it ever got past 10 to 12 percent. When the number stopped creeping up, that was the point at which they said, "Okay. What's going on here? We need to double back and make sure we have a world-class spreadsheet, and we need to diversify the company to create a set of other world-class applications that we'll eventually figure out how to make work together."

Tucker: That diversification also characterized most of the company. There were so few people who really knew the spreadsheet stuff—especially on the engineering team—and who once again had to completely reconstitute the team not just for Release 2, but then again for Release 3, because by then the Release 2 folks were off doing different things, whether that was Notes or Europa, which was the ESPD (Engineering and Scientific Products Division) spreadsheet. Europa was yet another team doing yet another spreadsheet, but they had taken people out of 1-2-3. There wasn't a good, solid continuous development process, in my view. There didn't appear to be—at the top levels of the company—a recognition that software development is important, that it's a process that drives the company and needs to be paid attention to. This core of engineers needed to be fed, watered, and nourished and it just didn't happen, at least not that way. So, Jon, again, you're the one who was there, and correct me if

you think that's wrong, but that was my observation, at pretty close range.

Sachs: I was only there during certain times, and a lot of what you're talking about happened after I left. But, let me just try to put you in the picture. There was some continuity in the teams. I did most of the work on 1-2-3, and then I was working on Symphony for a while before I kind of lost my stomach for it after the menus got like 18 levels deep.

Then we brought in other people to work on it, one of whom was Ray Ozzie, who I brought into the company. He used to work for me at Data General, then he went off to work for Software Arts for a while, and I hired him out of Software Arts. He made a deal with Mitch: Mitch said, "If you write the word processing component of Symphony, I'll let you do your own project next." So, he was in a real hurry to get that done. And the other person who was working on the communications part was Barry Spencer, who we had hired originally to write drivers for Version 1a. When we interviewed him, we thought, "This guy is a real dud, but we need people, so we'll just hire him anyway." And, he was really spectacular.

Tucker: He was stellar.

Sachs: A lot of the people who interviewed really great turned out to be really, really bad. Anyway, Barry became intimately familiar with the internals of 1-2-3 when he was working on the drivers. Then he worked on Symphony, and later he was the lead developer on Release 2 because he was one of the few who knew the assembly language code base for 1-2-3.

Tucker: But, meanwhile, he was the lead on Europa, and they had to pull him off that to get him on Release 2—or did that go the other direction?

Sachs: No, no, no. He did Europa after. There was another guy named Matt Stearns who came in when I was working on Symphony. He picked up some of the slack. He and Barry were the main developers who did Release 2, and I guess they both went to Europa. But there was a transition between when I was the only developer and after they started to ship 1-2-3. In the first place, I had told Mitch when I started I did not want to be V.P. of software development. I hate managing people; I get my enjoyment out of writing software and seeing it work. And he said, "That's fine. We'll hire people to do that." They hired two people out of Software Arts, who kind of jumped ship when Software Arts seemed to be lagging a little.

What management wanted, all of a sudden, was to do all these other versions of 1-2-3 and to start up other projects. They wanted to see stuff happen. For example, Symphony had to be translated into a lot of different languages because they wanted to get sales in Europe, so they just set up a whole office in England to do translations. And they had to set up a QA department—there were lots of things that had to be done.

So Dale and Dave were charged with building up software development but unfortunately, things don't work that way. What they were forced to do was hire a very large number of people in a very short amount of time. They brought in a recruiter named Sally Silver, who was a top recruiter, and they just did interviews one after the other. And because everything had to be done yesterday, they were only hiring experienced programmers—there was no entry-level hiring, no hiring out of school. So they basically hired people who came from other places, and hired out of a pool of available programmers which at any point in time is limited, and probably two-thirds of the people available at any point in time are available because they're not any good.

Tucker: And probably didn't have much PC experience.

Sachs: Yes. So, Dale and Dave were stuck with a tough job, suddenly having to hire all these teams. I think they did about the best they could; but, if you haven't lived through it, growth of 100 percent every six months is something you cannot imagine. It just creates unbelievable internal chaos. The people who were there at the beginning had a skill set adapted to working in a small, loose organization. They didn't work well in a larger organization, so they got kind of trampled. When you bring in a lot of people, they don't have time to learn how to work together, but everything somehow stumbles forward, and you quickly hire a number of people who aren't very good. They turn off the people who are good, and those people just want to get out of there. It was a really tough situation.

So they had a lot of trouble capitalizing on the development they had. More importantly, they never hired entry-level people or had an orientation program or mentored people—giving them time to grow within the organization and learn how to do things the same way everyone else was doing them. Without having actually been inside of Microsoft, I think that was more their model. You know, the Lotus middle management was very, very weak, and the structure never really pulled together.

Chronology of Product Releases

Tucker: Do you mind if I attempt a chronology—a sort of time and development team size and that sort of thing?

In 1982, there was principally one engineer—one developer on Lotus 1-2-3—you.

Sachs: There was another guy named George Reiner who wrote some of the conversion utilities.

Tucker: You and a couple of helpers.

Sachs: Yes.

Tucker: You had come to this having personally written several spreadsheets just before this, right? I mean, it's the Brooks theory—plan to throw one away; you will, anyway.

Sachs: Right. Plan to throw three away—yes.

Tucker: In terms of communication, there was nobody that you could communicate everything to, train and coordinate with, and so on. It all took place in your head. Then, Symphony—Symphony was a bigger team of core engineers. Three, four, something like that with more helpers and many more translation, QA, utilities Drivers, etc. Overall team size—I don't know; 30? Counting all those other people—but in terms of the core, maybe only three, four, or five?

Sachs: There was a whole group doing drivers for Version 1a in parallel. That was a team of five or six, and then there were a couple of people in management, and another three or four people floating around, not doing anything obvious.

Tucker: This would have been mid-1984. I remember using Symphony in February 1984. So, that's mid-1984 when Symphony shipped: July 1984. Then, shortly after that, a team was reconstituted for 1-2-3 Release 2, taking Symphony core code and in many cases stripping out stuff that wasn't needed anymore and turning it back into Lotus 1-2-3.

1-2-3 Release 2 was being prepared when I joined the company, which was July 1985. The product shipped around September 1985. And so that team had to be reconstituted again, but it was relatively small—it was a Symphonyesque team—three or four core developers and people doing support work.

Sachs: We're ignoring Jazz, which had like a 20-person team that was going on at the same time.

Kolowich: And No Comment, there's another one.

Tucker: Indeed. So, if 1-2-3 was the core of the company's revenues—$150 million to $200 million of revenues—it had two, three, or four people working on the core. Meanwhile, the company was growing all around it in whole new development efforts—a whole Mac team, a whole new platform team. And, Michael mentions No Comment, which was graphical. It became 1-2-3g; it was a big team, and those people wielded political power just through the sheer force of managing large groups of people. So, 1-2-3 Release 2 shipped. Then, that team split up. And there was a small engineering team, as we mentioned, on Europa. This would have been 1986. Europa was a 3D spreadsheet. Jon, you had written a 3D spreadsheet prototype, I recall. I

remember using it and seeing it.

Sachs: It was the last thing I did before I left. I said, “Gee, no one’s ever encapsulated the algorithms of a spreadsheet in a coherent form,” so I basically wrote a spreadsheet in C that kind of clarified how all this stuff works.

Kolowich: A schema of sorts?

Sachs: No, it was actually just a working program, a working spreadsheet. It wasn’t a complete product, but it was enough to show how the recalculation algorithms worked, how the compilers and decompilers and all that worked.

Tucker: When it came time to do a 1-2-3 Release 3—this is where I came into the picture, around late 1985; 1986 is when Lotus was seriously thinking about Release 3. The team had to be established from scratch because there was no longer a 1-2-3 Release 2 team. There were one or two members—Dan Sevush and Dennis Cunningham come to mind—who came over to the 1-2-3 Release 3 team, but Dan Sevush left not long after that. This team became enormous—50 people working on the product. Again, the product was written in C, by a team of people with relatively little experience personally implementing a spreadsheet. And, again, Jon came to the original task having implemented three or four personally, and the 1-2-3 Release 3 team was trying to do several things, one of which was build a spreadsheet for OS/2.

Dr. David Reid was in charge of the architecture and the development team, and I was in charge, ostensibly, of the feature set. David and I worked closely together, and I think the world of David, but he wasn’t experienced in the way that Jon was. The project grew very differently. After many months of us meeting, David and I had a conversation during which David said, “I thought my job was just to create another version of 1-2-3 for OS/2.” I said, “David, that’s not what my job is at all. My job is to create the next version of 1-2-3 with features.” He and I appeared to be working on just completely different projects.

Management Changes

Tucker: I always got the impression that Microsoft knew about development. They cared about it; it was deeply ingrained in their leader. Organizations take on the characteristics of the CEO, and it was around this time that Jim Manzi became the CEO and not Mitch. It was a very different atmosphere compared to a software-development-centric business. Lotus was not really a software development firm, in my opinion, after Jon left.

Campbell-Kelly: What was the date of the CEO change?

Kolowich: Mitch left in August 1986, but there were several things that happened before

that, over about a 10-month period. Jim Manzi stepped up to be President, I believe, in late 1984. Mitch went to run the Business Products Division. He was the general manager of that, working for Jim, who reported to Mitch.

Tucker: Meanwhile, Mitch was spending a lot of his time on a pet project.

Kolowich: Yes, for about three months, and that didn't work. So, at that point, Mitch, observing to some extent this breakdown of the development culture, said, "What I should really do is move my office over to 161 First Street, where a lot of the developers are, and really become the leader of the development organization." That was the situation for about three or four months, and then he left. So there was a rapid morphing of Mitch's role and Jim's role literally every three months over a year from mid-1985 to mid-1986.

Dramatic Growth and Its Resulting Problems

Morgan: As I look back on Lotus, it's this uncontrolled growth that caused all the problems the company eventually was to have, because when you grow as fast as that, you can't close the loop. You can't correct problems; you're out of control. It's like fungus growing on a Petri dish. When I left in 1987, all 40 or so people who were there when I joined in 1983 had left except for one person, I believe. It's a sort of classic start-up mind-set versus the next growth level in the company. But there were so many problems that we couldn't deal with adequately, and Mitch had a phrase that always stuck in my mind. He said one of the great challenges of any software company is insurmountable opportunity, and that was a problem: We couldn't grow fast enough.

There was trouble in paradise when we got our first mediocre reviews, and those were the Symphony reviews. And the company couldn't do more than become defensive at that point, and couldn't correct the problem because we were moving and changing so fast. I just kept seeing this over and over. I think all of you saw that. You see it in a lot of other companies that go through the same fast growth. I don't know what the lessons are, but as Jon said, unless you've been through that rapidly changing environment, you can't really appreciate it. It's craziness.

Kolowich: One of the key differences, for example, between what Microsoft was experiencing and what Lotus was experiencing was length of attention span and the ability to power through the setbacks driven by a strong long-term vision. You know, the first couple of versions of Windows were no great shakes. They were pretty pathetic.

Tucker: How about five years' worth of versions? I remember using the 1985 version, and Windows 3.0 was really the first usable version in 1990.

Morgan: When Microsoft sent us the first version of Windows, we just sat around and were laughing at it because it was so bad.

Kolowich: But there was a real commitment to, "Let's get it right. We'll learn from it, get it right, learn from it, get it right, learn from it, get it right." At Lotus, the instinct was, Okay, we've got an A list, a B list, and a C list. The A lists are the keepers. The B lists are on the bubble, and the C list projects got killed real fast.

Tucker: I would also say that there was the view that if you didn't get it perfect the first time then you were a loser. That's such a different point of view. And so, Jon, you screwed the company over because you got it perfect the first time. There really were unreasonable expectations.

Morgan: Mitch always used to say that most failures in software companies are suicides; they're not murders.

Release 3 Development Problems

Campbell-Kelly: You were telling us about the 1-2-3 Release 3 development. When did that process begin?

Tucker: It began in 1986. It was supposed to have been an 18-month effort to quickly get a spreadsheet out, so early 1988 would have been really nice. It didn't come out until late 1989.

Sachs: Actually, I have a footnote to add to that. I was back there as a consultant somewhere into the Release 3 project. Jim Manzi came by my office one day and said, "What do you think of the schedule?" because they were due to ship in six months or something. I said, "There is no way in hell you are going to ship that product anywhere near that time." And he said, "Oh, so you're willing to bet a case of wine?" I said, "Fine, anything." That was the last I ever heard.

Tucker: Jon, I don't know if you remember a conversation that you and I had. I sought you out at a certain point; I knew that you had come back to the company and I knew the history. I said, "Well, I'm involved in this project. You've been quoted, Jon, as saying that they're not going to ship on time; and, furthermore, it's going to be twice as slow and twice as big because they're doing it in C. This team is experienced and intelligent—why do you think that they're saying they can make it? Why do they think that they believe this and you believe that?" And your answer was "Denial. They're in denial." You were dead on.

Sachs: Well, I had no credibility because everyone thought I was just saying sour

grapes because it wasn't my project.

Tucker: Exactly. The political stuff was extraordinarily important at Lotus; it was very much about posturing and how many people reported to you and which office you had, and so on. That was not a good thing at all from a development standpoint.

Kolowich: Let's go back to the roots of Release 3. The whole Release 3 schedule and the decision not to go to Windows were both the seeds of Lotus's fall from prominence. If you go back, you can see that the Release 3 experience was based on lessons that the company thought it should learn from as far back as Symphony. Scott, you and I were talking earlier about the fact that Symphony actually was built on an assumption that turned out to be very fundamental. Even though Symphony was envisioned and sold as an upgrade from 1-2-3, it was not upward compatible with 1-2-3 on several important dimensions: on keystroke and macro support. Those were the two things.

The customer base raised all kinds of hell about that. At that time, Lotus was trying to build a large, professional mini-IBM-like sales organization for the large corporate customer base. They were working on very large site deals. Basically, the customers ended up beating up the salespeople over that. The corporate customers had a huge amount of influence by the time Release 3 was being designed so that the company felt it had to listen very carefully to every customer request that came over the transom. The features list, driven by corporate customers, kept growing and growing and growing. They also said, "Don't plan on upgrading us every 12 to 24 months. We only want to go through an upgrade under extraordinary circumstances, so make your upgrades big." It was: "Make it worth our while, and don't do it too often." In a lot of ways, the Release 3 experience came out of getting burned from the Symphony compatibility issue.

There was kind of a quick scramble to fix some of the perceived deficiencies in Release 1 to get to 2, so that was a pretty quick-shot release. But, Release 3 was a behemoth; and then, the combination of the massive feature list, the development team dynamics that we've been talking about, and a whole lot of things got in the way.

Tucker: There are other things, too. Another thing that the Release 3 team was trying to do was to invent a whole slew of technologies that would later be leverageable by lots of other groups within the company. Once again, it's the difference between approaches. Microsoft's vision of Windows was as a platform; they took a lot of time to think through and nurture and recognize that initial releases were not going to be good, and they took the time to get it right. With 1-2-3, we were trying also to make it into a system; yet, it had to be right and fast and perfect, right out of the gate the very first time. I agree with what Michael said—that many of the lessons that we tried to incorporate into the 1-2-3 Release 3 process, about understanding more of what the customer wants, the importance of compatibility and listening to what feature set the customers wanted—that was all true, and I think it all contributed.

You know, in some ways those lessons were learned at the wrong time. It would have been better if the lessons were learned a few years earlier and that there had been another release of 1-2-3. There was also the desire to have 1-2-3 be small and fast and compatible, but that's not the way it turned out. As a result, 1-2-3 Release 3 took so long that by the time it shipped, it missed the mark in terms of its feature set. By that time, the importance of laser printing, fonts, color, shading, and what we call spreadsheet publishing features were critically important. The 1-2-3 Release 3 had not been engineered to take advantage of those, so that got us into a whole separate series of technology investments, purchases, and so on.

Because 1-2-3 Release 3 had gotten so big, there was a segment of the hardware that it wouldn't run on. So, we had to do a Release 2.2 and then a 2.3. There was no 1-2-3 X team anymore. So, once again it had to be reconstituted, and we went to Matt and Barry for that code.

Of course, missing the bet on OS/2—Frank King had come to the company as number two dog or something. I remember a meeting with him where we put up our projections of what OS/2 penetration was going to be. Frank sat back and looked at it, and said, "Oh, it's going to be double that," and that was wildly optimistic in terms of how many people were going to switch. We missed a lot of targets and misread a lot of potential changes that were going to take place in the industry, and as a result of this long development, we basically hit the wrong target or missed the target.

Release 3 Features

Campbell-Kelly: Tell me a bit about the process of how this overrich feature set was decided upon.

Tucker: Well, I'm probably one of the chief contributors to that.

Sachs: Have you ever heard of the second system effect? I would say that this is the classic example.

Tucker: That's right. We collectively thought we knew what we were doing, and I was very much trusting in the development team to deliver. I would say, "This is what customers say they want;" the team would be saying, "Yes, we can do that."

Campbell-Kelly: Had you talked to customers?

Tucker: Oh, extensively. That's how I got drawn into 1-2-3 development. I mentioned before that I had worked for this aftermarket company, and so I knew the product inside out and backward, and Symphony as well. When I got my job at Lotus I thought that here was my

chance to really come to the oracle—here are the people who really know the products, and what I know is just the tip of the iceberg; now I'm really going to learn it. But no one knew anything about the products from my viewpoint. The product manager didn't know anything about the product and I was scared.

Sachs: Yes, I remember a meeting that Frank King attended. I said something about macros, and he asked me, "What's a macro?" I was so appalled, I could not believe it. I was just shaking my head the rest of the day.

Tucker: There was so much to know about Lotus, about the whole company—strategies, diversification, markets, different projects, and so on—that it was really hard to concentrate on just the one thing that really drove the whole company: the spreadsheet and Lotus 1-2-3 itself. So I got drawn into this, and it was like, all right, I have this important role for which I really felt I was ill-prepared, but I did the best I could because it didn't seem like anybody else knew what they were doing, either. I visited an awful lot of customers. As Michael pointed out, blending in customer feedback was a key part of what 1-2-3 Release 3 was doing. We held what we called product councils and alpha reviews, and I think we did the company a big favor in many ways by institutionalizing this process of getting customer feedback, doing usability testing, and so on. But the team was so far behind, and the project got so big and bloated, that nothing could save that particular project.

Kolowich: I thought the Release 3 spec was a great articulation of what customers wanted in 1985. Yet it couldn't be implemented for three years, at which point the customer needs changed, and the platform had changed, and the expectations had changed.

Tucker: It's just as important to decide what you're not going to do as what you're going to do, and that didn't come from the project team because – well, one can speculate why. But a seasoned, experienced, cohesive group of people didn't exist like the group that had previously implemented versions of this same product. It's certainly a foreseeable outcome in hindsight.

Lotus 1-2-3 as a Platform

Campbell-Kelly: You had told me on a previous occasion that there was a potential in Lotus 1-2-3 for it to become the operating system platform for the PC. I'd like you to tell us what you really meant by that, and if other people shared that vision.

Tucker: That's a big topic, but I think it was recognized at the time. First of all, Microsoft then wasn't what it is today. Microsoft had DOS. There was a lot of discussion about windowing environments and the like, and it wasn't at all clear that Windows was going to be the successful dominant platform. Lotus was very eager not to let Microsoft take that business, yet Lotus didn't really have a product, or a suitable product, to do that. We were reaching.

Kolowich: The other thing, though, is that Microsoft had shown its hand with Excel for the Mac at that point, which was essentially trouncing Jazz.

Tucker: With so many businesses relying on Lotus and with the macro capability that existed, a lot of automation was taking place. I think we all wanted to believe that somehow 1-2-3 could become the type of enterprise system software that Windows has certainly become. In fact, Microsoft is currently marketing Office as that type of system, and they have a reasonable chance of doing that. But, it just shows how long it's taken, and I'm not even sure that Microsoft is going to be successful in doing anything other than entrenching its market share of Office. I don't know that it's ever going to be a platform, per se.

But these discussions were key to so many things. And, Martin, yes, we've spoken quite a bit about the aftermarket. I must have contributed at Lotus to a great deal of this direction, too, having come from one of the only seemingly legitimate aftermarket firms into this role. At my first job, part of my job description was to make the worksheet file format the de facto standard for tabular information interchange. I remember being around to publish the WKS and WK1 formats at the time, and Symphony as well. We were very much involved in getting developers on board, and that contributed to the platform. Now, looking back, how many really successful aftermarket companies were there? Almost none—maybe none.

Tarter: Why was that, Scott?

Tucker: A couple of reasons. One is the really valuable additions to the product—features that should have been in the product to begin with or would shortly be added. Being able to print great-looking documents—there were a couple more. One of them, think, allowed you to see it at a different resolution. That seems quaint now. There were a few others, but the most successful ones that appealed to a broad enough base were features that were eventually subsumed by the product itself, and those that were very specific had a small enough market share that appealed to such a small group of people that it was unsustainable.

Morgan: It's interesting. I have seen that model work. If you look at PhotoShop, the plug-in aftermarket for PhotoShop seems to be doing fine.

Kolowich: Well, there's nowhere near the money in the PhotoShop plug-in market that's in the PhotoShop market, speaking as a PhotoShop plug-in author.

Tucker: I've been in this market ever since, by the way. From the day that we started the company, we said we were not going to be the prototypical third-party developer trying to make a go of it because that's not a viable market. What we were going to do instead was sell to OEMs. We were going to find companies that saw value in what we did on a one-off basis and sell to them a large contract for something that may add onto Office, and we have found that

model to be successful. But, even today, count the number—can any of us name an Office add-on vendor? No.

Tarter: In fairness, however, I remember the first Lotus developers' conference. The add-on people weren't necessarily the big source of energy there. What surprised everybody was the number of corporate developers who turned out. Something like 80 percent of the attendees were developing internal applications, which would argue that this notion of it being a platform was not entirely crazy. There really was enthusiasm for using it for internal purposes.

Tucker: I couldn't agree more. I would say that that is exactly the direction that was most important—to pay attention to how he spreadsheet could be better integrated into business practices in any given organization as opposed to it being a generalized platform so that other software developers could make it into a business model. The key issue here is: was it a business model? It really wasn't.

Sachs: I take some personal responsibility for that. The macro language was exactly that. It was a set of keystrokes, and it wasn't a general-purpose language. I warned Mitch that we could do this, and it would be really quick and dirty and powerful, but that there was going to be a price to be paid down the line. When you tie everything to keystrokes, you can't change the menus, you can't change anything. Yes, you could automate things and write little programs in it that do neat things, but it wasn't something you could build on, and it wasn't something that had a future. A number of times, I played with the idea of a more general-purpose language that could be embedded in a spreadsheet. But then a lot of end users couldn't have used it because all of a sudden if you've got programming, that scares everybody, whereas keystrokes made it accessible. To make that transition, they would have had to incorporate some general-purpose language within the product.

Tucker: Which Microsoft did brilliantly in VBA – Visual Basic; it is a very elegant implementation. And, it's visual. VBA is Visual Basic for all intents and purposes, and it drives Excel and the other Office applications beautifully. But, look at how much effort and engineering went into that. It is much more than, "Hey, we could add keystrokes or even make some commands in curly braces" and so on. I first demonstrated a set of business applications in New Orleans at Softcon in February 1984. We had hacked the worksheet file and been able to get both high-order and low-order characters into the spreadsheet itself. We had discovered just through pure hacking that the macro commands—the macro keystroke, curly brace things—could be encapsulated by one character. It was a real trick to try to get the character into the cell. Essentially, we had this kind of compiled version of macros, and then we'd use the higher order of characters for graphics around our things and so on. And you must have been thinking, "This is exactly what I was afraid of" because that's what we had done. That was maybe the beginning of the end.

What Went Wrong at Lotus

Sachs: I'm sensing an undercurrent here of "what went wrong with Lotus." They were flying so high, then they ended up—well, they didn't crash, but they turned into something else before they were scheduled to crash. There were some key problems if you contrast Lotus and Microsoft, and one of these was brought up earlier. We were talking about Jim Manzi being in charge, and 1-2-3 Release 3. Jim Manzi wasn't entirely popular with a lot of people, but I had a lot of sympathy for him because he wasn't a technical person at all. He was a journalist, and then he was a management consultant with McKinsey, and that's how he met Mitch because they were using them as a management consultant for a while. He didn't really know anything about software, and he didn't really care anything about software.

I don't think that was a problem except when he hired Dave Reid, who was in charge of the Release 3 project. Jim hired Dave from MIT, and he was a very smart person with no practical experience developing software in industry. He had ridiculously optimistic estimates of how long things would take, and he just hadn't been in the trenches. Jim didn't know enough to know that he was getting fed a line of bull; Dave wouldn't have lasted ten minutes in front of Bill Gates. Gates would have thrown him out saying, "Look, this is ridiculous. You can't do it in that amount of time."

There was another really huge mistake that Lotus made which has to do with losing the suite market. They won the spreadsheet market, but they lost the battle—nobody just buys a spreadsheet anymore; they buy suites. The way Lotus dealt with suites was they bought other companies. They bought AmiPro as a word processor, and they bought Freelance's graphics and presentations. They never brought all the development teams into one place, beat them about the head until they merged everything, made it look like one product and made everything work together, I'm sure there were meetings where Bill Gates read his guys the riot act and said, "Look, your menus don't match up. This doesn't look like an integrated product. It looks like a bunch of things glued together." Lotus never got past having individuals working on projects in different geographical locations. There wasn't anyone at the top who grasped the problem of "This isn't working, guys; we've got to really make this all look like one thing."

Tucker: The animosity that Lotus had toward Microsoft—and therefore toward Windows and everything graphical, platform, and consistency—hurt the process of being able to come up with a suite because the suite only makes sense if the applications can all work similarly. Well, they work similarly if they're all following a style guide of a platform. One of my observations about what happened at Lotus was that because 1-2-3 Release 3 was late, we were reduced to arguing to customers: "Customers want the 1-2-3 interface. Look at all the satisfied people who are out there. They keep telling us they want compatibility. That's what we're giving them. It's not some theoretically good interface; it's a proven good interface, and this is what people want." That really is what we were telling customers, but we came to believe it ourselves, or at least some of us did. The reality is that we should have been all over graphical interfaces, and

we should have been learning the lessons of the importance of simplicity, consistency, and so on. These were lessons that were obvious on the Mac.

Kolowich: Well, to be fair, pretty early on there was a reasonably comprehensive graphical interface development program oriented toward a suite that we mentioned earlier. The code name for the whole project was No Comment. It had a word processor called Allegro, a database called Baseline, and a spreadsheet called Vertigo—again, internally.

That felt like a long-term-oriented project. It was somewhat segregated—they had their own building; they had a pretty good team. The issue, though, was that Jim, for corporate strategy reasons, said, “This product will never ship on Windows.” In fact, he did make a declaration that this company will never ship a Windows product. I had responsibility in my group for the Notes project outside, and he instructed me to tell Ray Ozzie to stop work on Windows and make sure that Notes was an OS/2 product.

So, there was a whole corporate strategy overlay there that basically said we will not do this on Windows, but by the time it became clear that OS/2 wasn’t going anywhere and Windows was, it was much too late to restart the No Comment project. They saved the spreadsheet version, and that became 1-2-3G, but they couldn’t convert everything else. A graphical suite had to be put together by cobbling together AmiPro for the word processor and the like, but it just wasn’t designed to work that way.

But there was a development team with very good intentions to build a suite-oriented product. They had a three-year time horizon to design it, to try to do it right, and they just missed the platform.

Sachs: I was actually a consultant working on 1-2-3G at that time, so, I saw that from the inside as well. But the final blow was that Microsoft then was encouraging everyone to develop a presentation manager for OS/2. They said, “This is the future,” so it was like their Symphony.

Tucker: Yes, exactly.

Sachs: And, part way through, IBM dragged Microsoft down in endless bureaucracy and compatibility issues, and then they overshot the 8086 platform. And some rogue group within Microsoft said, “Hey, we can fix Windows 3,” and they solved the memory problem so it worked with some kind of kluged-up memory management so you could have large programs. All of a sudden it took off, and everyone else got left in the dust except for Microsoft who had to develop for Windows 3 because it was their product. It would have been an embarrassment if they didn’t have versions of Excel and Word. I’m convinced that it was completely unintentional on Microsoft’s part that they basically dealt a devastating blow to almost everyone else in the

industry.

Kolowich: That is true. A lot of people do ascribe motive to it, that there was this elaborate head fake. I remember Steve Ballmer coming in and saying that, because of memory limitations, the Windows path is a dead-end; our commitment is to OS/2 presentation manager.

Sachs: Right. I don't think they were being disingenuous.

Tucker: Agreed. Agreed.

Kolowich: For the record, there is a variety of opinions on that.

Tucker: Sure, but we did happen to have an inside view, and IBM was very involved in the decisions that Microsoft was making. And, it certainly appeared at the time that OS/2 and OS/2 PM was the way to go. But that doesn't diminish the fear or whatever you want to call it—the distaste—for doing anything oriented to Windows. I don't know that it would have been any better for the company to have supported one outside platform versus another. Whether it was IBM's or Microsoft's I don't think would have made that much difference. But there was this personal animosity toward Microsoft.

Kolowich: I'm not sure whether the course of history would have gone exactly the same way if OS/2 had survived, because I think Lotus was much better prepared for that path.

Tucker: That's a fair comment, and thank you for reminding us about No Comment and that development effort.

Morgan: As an ancillary comment: At the time I was consulting to several software companies, and they all were astonished at this juggernaut that came out of IBM to have them develop OS/2. I had never seen anything like it; they really tried to force it down the throats of developers everywhere, and it just failed completely. But it was an amazing attempt—unforgettable in its intensity.

Tucker: There are a lot of interesting parallels, I would say. Jon brought up the issue that in some ways OS/2 Presentation Manager was Microsoft's Symphony. There was a transition in terms of the development process from a relatively small internal group doing Windows changing over to this global team—Hursley, England, wasn't it, where half the presentation manager developers were? It was typical IBM, it was enormous and slow—and complicated—and so was the product.

Competitors

Haigh: On each of these points—with Lotus 1-2-3 Release 1, Symphony, Lotus 1-2-3 Release 2, Lotus 1-2-3 Release 3, and these other products you've been talking about—what were the perceived competitors for each of these products? How was this sense of what niche each product was going into arrived at, and what were the competitors doing that shaped these decisions about features you needed and how the product should be produced and sold? The contrast with VisiCalc has been made many times, and we've already talked about the speed, the macros, which obviously were made for PC support, but then there were a bunch of other spreadsheets out there as well.

Sachs: Well, the fact is that within a month or two of when 1-2-3 first shipped, it hit the top of the charts and stayed there for three years or so as the top-selling product. People tried to compete, but basically Lotus had 90-some percent of the market.

Morgan: What about Context MBA?

Sachs: When 1-2-3 was first released, there was a little bit of dithering when some articles compared it with Context MBA. Context MBA should have been a lesson because that was like Symphony. It had five functions. It had communications. It had a bunch of other things. But, the fatal flaw that they made in getting to market quickly—and they actually got to market before 1-2-3—was they implemented Context MBA using UCSD Pascal, which was interpretive. Forget that you're writing in C; this was another factor of ten slower than writing in C. It let them get something to work very quickly, but it was unbelievably slow and large. So, basically it became a footnote in history.

Haigh: And, VisiCorp's VisiOn?

Sachs: It was demoware that never quite materialized. There was a lot of flash about that.

Kolowich: Other ones that popped up on the horizon were the two knockoffs called Mosaic and Twin—and then Borland's quasi-knockoff, which was Quattro Pro. Basically Lotus just treated those as legal defense issues; that's a whole other thread, the copyright defense. But when they came on the market, there were a lot of people who were interested. There was one big corporate account; GE actually made a contract, I believe, with one of them—basically, GE bought an unlimited contract, I believe, for Twin for $50,000 or something like that for as many copies as they wanted. But, much to GE's amazement, people were still buying 1-2-3 off the books, off the contract. But Lotus chose to beat them down by aggressively pursuing the copyright action. The little guys went away quickly, and Borland decided that they would change Quattro enough so that they would stand a chance in court of being able to survive the legal

issues.

Tucker: It's a bit of an orthogonal thought, but this issue about clones was fascinating. I had the assignment while at Optionware of being a compatibility consultant to the clone makers. And then I turned around and become the design manager of 1-2-3. Then I was deposed in the lawsuits that Lotus was prosecuting against the companies that I had previously consulted for—it was an amusing situation to be in.

Haigh: So, other than the legal steps necessary to protect the 1-2-3 look and feel, and other intellectual property quirks, there wasn't really a worry in the company that there were credible competitors so that there needed to be a new version at this point with these features? You were really just operating almost in a vacuum when it came to new features?

Tucker: It depends on the chronology. At the time that Jon is talking about, he's absolutely right. 1-2-3 dominated, they were out very quickly in the market, and that was that. Now, as for later in the company's history, yes, we were quite concerned about all these various clones, and we were concerned about things that we were afraid we'd never heard of that might be lurking around the corner. We were very concerned about Microsoft, you know.

Sachs: Yes, Excel. Excel was scary.

Tucker: It was very scary. What Microsoft had going for it, as much or more than anything else, was the platform. They had the platform, they had a great spreadsheet in the platform, and we were too focused on compatibility and on making this thing the same as what people were used to. We didn't recognize that when there's a paradigm shift, that's the time where you can make a compatibility break. Microsoft took advantage of that, and, we could have but missed it, for the reasons that Michael has brought up. I remember talking with Bill Gates at a Comdex dinner for a good half-hour, and he knew quite a bit about Lotus's organization. He knew who I was and who my manager was and who he reported to, and so on—he knew way too much for my comfort about our company and about 1-2-3 and our plans. We got into a little bit of a debate about whether Windows was an operating system or an environment or was it a platform; and, well, wasn't 1-2-3 a platform? His parting comment to me was, "Well, we'll let the market decide."

