ACCIDENTAL EMPIRES

CHAPTER FIFTEEN

**FUTURE COMPUTING**

Remember *Pogo?*Pogo was *Doonesbury*in a swamp, the first political cartoon good enough to make it off the editorial page and into the high-rent district next to the horoscope. Pogo was a ‘possum who looked as if he was dressed for a Harvard class reunion and who acted as the moral conscience for the first generation of Americans who knew how to read but had decided not to.

The *Pogo*strip remembered by everyone who knows what the heck I am even talking about is the one in which the little ‘possum says, “We have met the enemy and he is us.” But today’s sermon is based on the line that follows in the next panel of that strip—a line that hardly anyone remembers. He said, “We are surrounded by insurmountable opportunity.”

We *are*surrounded by insurmountable opportunity.

Fifteen years ago, a few clever young people invented a type of computer that was so small you could put it on a desk and so useful and cheap to own that America found places for more than 60 million of them. These same young people also invented games to play on those computers and business applications that were so powerful and so useful that we nearly all became computer literate, whether we wanted to or not.

Remember computer literacy? We were all supposed to become computer literate, or something terrible was going to happen to America. Computer literacy meant knowing how to program a computer, but that was before we really had an idea what personal computers could be used for. Once people had a reason for using computers other than to learn *how*to use computers, we stopped worrying about computer literacy and got on with our spreadsheets.

And that’s where we pretty much stopped.

There is no real difference between an Apple II running VisiCalc and an IBM PS/2 Model 70 running Lotus 1-2-3 version 3.0. Sure, the IBM has 100 times the speed and 1,000 times the storage of the Apple, but they are both just spreadsheet machines. Put the same formulas in the same cells, and both machines will give the same answer.

In 1984, marketing folks at Lotus tried to contact the people who bought the first ten copies of VisiCalc in 1979. Two users could not be reached, two were no longer using computers at all, three were using Lotus 1-2-3, and three were still using VisiCalc on their old Apple IIs. Those last three people were still having their needs met by a five-year-old product.

Marketing is the stimulation of long-term demand by solving customer problems. In the personal computer business, we’ve been solving more or less the same problem for at least ten years. Hardware is faster and software is more sophisticated, but the only real technical advances in software in the last ten years have been the Lisa’s multitasking operating system and graphical user interface, Adobe’s PostScript printing technology, and the ability to link users together in local area networks.

Ken Okin, who was in charge of hardware engineering for the Lisa and now heads the group designing Sun Microsystems’ newest workstations, keeps a Lisa in his office at Sun just to help his people put their work in perspective. “We still have a multitasking operating system with a graphical user interface and bit-mapped screen, but back then we did it with half a mip [one mip equals one million computer instructions per second] in 1 megabyte of RAM,” he said. “Today on my desk I have basically the same system, but this time I have 16 mips and an editor that doesn’t seem to run in anything less than 20 megabytes of RAM. It runs faster, sure, but what will it do that is different from the Lisa? It can do round windows; that’s all I can find that’s new. *Round windows,*great!”

There hasn’t been much progress in software for two reasons. The bigger reason is that companies like Microsoft and Lotus have been making plenty of money introducing more and more people to essentially the same old software, so they saw little reason to take risks on radical new technologies. The second reason is that radical new software technologies seem to require equally radical increases in hardware performance, something that is only now starting to take place as 80386- and 68030-based computers become the norm.

Fortunately for users and unfortunately for many companies in the PC business, we are about to break out of the doldrums of personal computing. There is a major shift happening right now that is forcing change on the business. Four major trends are about to shift PC users into warpspeed: standards-based computing, RISC processors, advanced semiconductors, and the death of the mainframe. Hold on!

\*\*\*\*\*\*\*\*\*\*

In the early days of railroading in America, there was no rule that said how far apart the rails were supposed to be, so at first every railroad set its rails a different distance apart, with the result that while a load of grain could be sent from one part of the country to another, the car it was loaded in couldn’t be. It took about thirty years for the railroad industry to standardize on just a couple of gauges of track. As happens in this business, one type of track, called*standard gauge,*took about 85 percent of the market.

A standard gauge is coming to computing, because no one company—even IBM—is powerful enough to impose its way of doing things on all the other companies. From now on, successful computers and software will come from companies that build them from scratch with the idea of working with computers and software made by their competitors. This heretical idea was foisted on us all by a company called Sun Microsystems, which invented the whole concept of open systems computing and has grown into a $4 billion company literally by giving software away.

Like nearly every other venture in this business, Sun got its start because of a Xerox mistake. The Defense Advanced Research Projects Agency wanted to buy Alto workstations, but the Special Programs Group at Xerox, seeing a chance to stick the feds for the entire Alto development budget, marked up the price too high even for DARPA. So DARPA went down the street to Stanford University, where they found a generic workstation based on the Motorola 68000 processor. Designed originally to run on the Stanford University Network, it was called the S.U.N. workstation.

Andy Bechtolscheim, a Stanford graduate student from Germany, had designed the S.U.N. workstation, and since Stanford was not in the business of building computers for sale any more than Xerox was, he tried to interest established computer companies in filling the DARPA order. Bob Metcalfe at 3Com had a chance to build the S.U.N. workstation but turned it down. Bechtolscheim even approached IBM, borrowing a tuxedo from the Stanford drama department to wear for his presentation because his friends told him Big Blue was a very formal operation.

He appeared at IBM wearing the tux, along with a tastefully contrasting pair of white tennis shoes. For some reason, IBM decided not to build the S.U.N. workstation either.

Since all the real computer companies were uninterested in building S.U.N. workstations, Bechtolscheim started his own company, Sun Microsystems. His partners were Vinod Khosla and Scott McNealy, also Stanford grad students, and Bill Joy, who came from Berkeley. The Stanford contingent came up with the hardware design and a business plan, while Joy, who had played a major role in writing a version of the Unix operating system at Berkeley, was Mr. Software.

Sun couldn’t afford to develop proprietary technology, so it didn’t develop any. The workstation design itself was so bland that Stanford University couldn’t find any basis for demanding royalties from the start-up. For networking they embraced Bob Metcalfe’s Ethernet, and for storage they used off-the-shelf hard disk drives built around the Small Computer System Interface (SCSI) specification. For software, they used Bill Joy’s Berkeley Unix. Berkeley Unix worked well on a VAX, so Bechtolscheim and friends just threw away the VAX and replaced it with cheaper hardware. The languages, operating system, networking, and windowing systems were all standard.

Sun learned to establish de facto standards by giving source code away. It was a novel idea, born of the Berkeley Unix community, and rather in keeping with the idea that for some boys, a girl’s attractiveness is directly proportional to her availability. For example, Sun virtually gave away licenses for its Network Filing System networking scheme, which had lots of bugs and some severe security problems, but it was free and so became a de facto standard virtually overnight. Even IBM licensed NFS. This giving away of source code allowed Sun to succeed, first by being the standard setter and then following up with the first hardware to support that standard.

By 1985, Sun had defined a new category of computer, the engineering workstation, but competitors were starting to catch on and catch up to Sun. The way to remain ahead of the industry, they decided, was to increase performance steadily, which they could do by using a RISC processor—except that there weren’t any RISC processors for sale in 1985.

RISC is an old IBM idea called Reduced Instruction Set Computing. RISC processors were incredibly fast devices that gained their speed from a simple internal architecture that implements only a few computer instructions. Where a Complex Instruction Set Computer (CISC) might have a special “walk across the room but don’t step on the dog” instruction, RISC processors can usually get faster performance by using several simpler instructions: walk-walk-step over-walk-walk.

RISC processors are cheaper to build because they are smaller and more can be fit on one piece of silicon. And because they have fewer transistors (often under 100,000), yields are higher too. It’s easier to increase the clock speed of RISC chips, making them faster. It’s easier to move RISC designs from one semiconductor technology to a faster one. And because RISC forces both hardware and software designers to keep it simple, stupid, they tend to be more robust.

Sun couldn’t interest Intel or Motorola in doing one. Neither company wanted to endanger its lucrative CISC processor business. So Bill Joy and Dave Patterson designed a processor of their own in 1985, called SPARC. By this time, both Intel and Motorola had stopped allowing other semiconductor companies to license their processor designs, thus keeping all the high-margin sales in Santa Clara and Schaumberg, Illinois. This, of course, pissed off the traditional second source manufacturers, so Sun signed up those companies to do SPARC.

Since Sun designed the SPARC processor, it could buy them more cheaply than any other computer maker. Sun engineers knew, too, when higher-performance versions of the SPARC were going to be introduced. These facts of life have allowed Sun to dominate the engineering workstation market, as well as making important inroads into other markets formerly dominated by IBM and DEC.

Sun scares hardware and software competitors alike. The company practically gives away system software, which scares companies like Microsoft and Adobe that prefer to sell it. The industry is abuzz with software consortia set up with the intention to do better standards-based software than Sun does but to sell it, not give it away.

Sun also scares entrenched hardware competitors like DEC and IBM by actually encouraging cloning of its hardware architecture, relying on a balls-to-the-wall attitude that says Sun will stay in the high-margin leading edge of the product wave simply by bringing newer, more powerful SPARC systems to market sooner than any of its competitors can.

DEC has tried, and so far failed, to compete with Sun, using a RISC processor built by MIPS Computer Systems. Figuring if you can’t beat them, join them, HP has actually allied with Sun to do software. IBM reacted to Sun by building a RISC processor of its own too. Big Blue spent more on developing its Sun killer, the RS/6000, than it would have cost to buy Sun Microsystems outright. The RS/6000, too, is a relative failure.

\*\*\*\*\*\*\*\*\*\*

Why did Bill Gates, in his fourth consecutive hour of sitting in a hotel bar in Boston, sinking ever deeper into his chair, tell the marketing kids from Lotus Development that IBM would be out of business in seven years? What does Bill Gates know that we don’t know?

Bill Gates knows that the future of computing will unfold on desktops, not in mainframe computer rooms. He knows that IBM has not had a very good handle on the desktop software market. He *thinks*that without the assistance of Microsoft, IBM will eventually forfeit what advantage it currently has in personal computers.

Bill Gates is a smart guy.

But you and I can go even further. We can predict the date by which the old IBM—IBM the mainframe computing giant— will be dead. We can predict the very day that the mainframe computer era will end.

Mainframe computing will die with the coming of the millennium. On December 31,1999, right at midnight, when the big ball drops and people are kissing in New York’s Times Square, the era of mainframe computing will be over.

Mainframe computing will end that night because a lot of people a long time ago made a simple mistake. Beginning in the 1950s, they wrote inventory programs and payroll programs for mainframe computers, programs that process income tax returns and send out welfare checks—programs that today run most of this country. In many ways those programs have become our country. And sometime during those thirty-odd years of being moved from one mainframe computer to another, larger mainframe computer, the original program listings, the source code for thousands of mainframe applications, were just thrown away. We have the object code—the part of the program that machines can read—which is enough to move the software from one type of computer to another. But the source code—the original program listing that people can read, that has details of how these programs actually work—is often long gone, fallen through a paper shredder back in 1967. *There is mainframe software in thiscountry that cost at least $50 billion to develop for which no source code exists today.*

This lack of commented source code would be no big deal if more of those original programmers had expected their programs to outlive them. But hardly any programmer in 1959 expected his payroll application to be still cutting checks in 1999, so nobody thought to teach many of these computer programs what to do when the calendar finally says it’s the year 2000. Any program that prints a date on a check or an invoice, and that doesn’t have an algorithm for dealing with a change from the twentieth to the twenty-first century, is going to stop working. I know this doesn’t sound like a big problem, but it is. *It’s a very big problem.*

Looking for a growth industry in which to invest? Between now and the end of the decade, every large company in America either will have to find a way to update its mainframe software or will have to write new software from scratch. New firms will appear dedicated to the digital archaeology needed to update old software. Smart corporations will trash their old software altogether and start over. Either solution is going to cost lots more than it did to write the software in the first place. And all this new mainframe software will have one thing in common: it won’t run on a mainframe. Mainframe computers are artifacts of the 1960s and 1970s. They are kept around mainly to run old software and to gladden the hearts of MIS directors who like to think of themselves as mainframe gods. Get rid of the old software, and there is no good reason to own a mainframe computer. The new software will run faster, more reliably, and at one-tenth the cost on a desktop workstation, which is why the old IBM is doomed.

“But workstations will never run as reliably as mainframes,” argue the old-line corporate computer types, who don’t know what they are talking about. Workstations today can have as much computing power and as much data storage as mainframes. Ten years from now, they’ll have even more. And by storing copies of the same corporate data on duplicated machines in separate cities or countries and connecting them by high-speed networks, banks, airlines, and all the other other big transaction processors that still think they’d die without their mainframe computers will find their data are safer than they are now, trapped inside one or several mainframes, sitting in the same refrigerated room in Tulsa, Oklahoma.

Mainframes are old news, and the $40 billion that IBM brings in each year for selling, leasing, and servicing mainframes will be old news too by the end of the decade.

There is going to be a new IBM, I suppose, but it probably won’t be the company we think of today. The new IBM *should*be a quarter the size of the current model, but I doubt that current management has the guts to make those cuts in time. The new IBM is already at a disadvantage, and it may not survive, with or without Bill Gates.

http://www.cringely.com/2013/03/28/accidental-empires-chapter-15-future-computing/