Teresa Jennings

From: Nathan Myhrvold
To: Bill Gates; Peter Rinearson (Alki Software); Jonathan Lazarus
Subject: ARTICLE.DOC
Date: Saturday, July 24, 1993 11:56AM

This is a draft of the memo I mentioned in the meeting the other day with Peter. It is an article about the software business model etc. Whether this is valuable for the book, or really for an article or whatever we'll have to see but I thought I'd send it on to you guys to take a look.

Nathan

<<File Attachment: ARTICLE.DOC>>

InterOffice Memo

To: List

From: Nathan P. Myhrvold

Date: July 18, 1993

Subject: **Telling It Like It Is**

This memo was born out of frustration. I was thinking this weekend about a variety of things involving our public image, how our adversaries use public opinion against us, and how we respond. Here are some particular things that make me mad:

- **Competitors routinely use innuendo and vague claims of wrong doing against us.** So far we have basically not responded, or if we have, the message does not come across anywhere near as well as theirs does. We do not have a very articulate response.
- **Random assholes-on-a-soapbox put forward their pet theories at our expense.** Mitch Kapor is one such person - he will attempt to use his new found lobbying power against us. Self appointed "experts" like Charles Ferguson attempt to become the pundit de jour by writing all sorts of crap about how we plotted IBM's downfall (like they needed help doing this!). Besides being annoying, I wind up having to combat this bullshit with every company that we do business with. The current cable deal is *full* of terms and structures motivated directly by these claims.
- **Even when we do talk, our message doesn't get out.** In the last two years I have given my little "Windows as a Scalable Architecture" speech dozens of times to reporters, analysts etc. There is a well worn path in my office (I sometimes call it the Waggoner-Edstrom Trail) where these people come to watch me gesticulate for an hour. None of the message gets out however - except for a few random sound bites on other topics. I guess the good news is that I can keep the same gig going for years before anybody notices the repetition. I'm not mad about doing my little dance, but it does strike me that there may be more effective ways of getting a message like this out.

Normally when these things piss me off (after all, they are not new) I am able to suppress the thoughts with a variety of different antidotes. There is the "cynical and worldly wise" where I shrug it off as something to be expected. There is the "crying all the way to the bank" antidote where I console myself with our success. There is the "high road" antidote where I tell myself that really it is better not to stoop to their level.

This weekend I decided to take a different approach. I think that we need to get some of our side of the story out. It might be momentarily satisfying to do this in the way that these other folks do, but I think that we can do even better in another forum.

The method that I chose was to write a sample of a magazine article which discusses the fundamental business dynamics of the computer industry from our perspective. This is our chance to "tell it like it is" - at least to a certain degree. Perhaps the *Harvard Business Review*, would work because it is willing to take on complicated issues in depth. If HBR doesn't work, then perhaps this could be reworked for another business magazine.

The draft article below may be terribly worded, is too long, and may miss the mark so don't take it literally - I just pounded it out without much editing. It's written in the first person, which might be all wrong.

The basic goals that I think the *real* article should have are:

- **Authored by an MS exec.** This gives the thing credibility. The strongest statement would be to have BillG author it, or he co-author. Another strategy would be to have somebody else do it

because lower rank voices can sometimes get away with being a bit more strident. Note that I have written the draft below as if I was the author (because so far, I am!), but I am not insisting that this be the case.

- **Take on the "monopoly" issue directly.** The story of MS Dos is something to be proud of, and there is both business and academic issues to point in explaining it. There is clearly a risk that we will only exacerbate the issue, but I think that it is worth doing.
- **Explain why we're winning and others are losing.** Imagine that you're some relatively smart person who understands business, but doesn't know our industry in detail - i.e. you're in the press, or at another company considering doing business with us, or you're a congressional staffer... A person like that is very vulnerable to stories that we have somehow colluded to create a monopoly, or other dastardly deeds, because our level of success is so incredibly high.
- **Point out a few emperors with no clothes.** The goal is not to bash people, but I don't see any reason not to draw on some examples from the industry to illustrate our point.

Here is my first cut at something that might do this, albeit poorly. Even if we don't use this anywhere it has been cathartic to write it!

The Economic Dynamics of Software

Computer technology continues to revolutionize the ability for individuals to capture, create and manipulate information. The personal computing revolution has spread this technology beyond the rarefied confines of the ivory tower, research lab or corporate computing center to directly touch the lives of millions of office and information workers. It has become increasingly clear that the next wave of this revolution will engulf the world of consumer information and become the new medium for a whole host of information services from communication to entertainment. Indeed, many predict that even the most personal sort of intimacy will be mediated (or simulated) by computer driven virtual reality technology.

In the process, the computer industry has become the bellwether of American business and a leitmotif for high technology industries in general. It is held up by politicians and pundits as an architypical model for the economic value of high technology, American competitiveness in the world economy and a ray of hope for the future. Despite this great interest and scrutiny, I find that most accounts miss the point, and the fundamental economic dynamics of the computer industry are still widely misunderstood .

I must confess that I am *not* an impartial observer. I am a senior vice president at Microsoft, and over the last seven years I've worked on our technical and business strategy, both for existing products and new lines of business in the larger realm of digital information. The purpose of this article is to discuss some of these issues from the point of view of a practitioner in this business.

The Monopoly that Isn't

The fact that the VHS video tape format prevailed over the Beta format used by the Sony is probably familiar to most of us. Although Beta started first and was championed by Sony, it wound up losing the battle so thoroughly that players for it are no longer even made. As a result, VHS currently has virtually 100% of the market share for home players and prerecorded video tapes (8mm video tape is a contender, but so far only for handheld camcorders). Classical economics would tell us that this is has all the earmarks of a monopoly. One would expect that the Japan Victor Company which controls and licenses the VHS standard to be reaping monopoly profits, and that both consumers and competitors would be suffering the various harms of a monopoly - exorbitant pricing, restrains on free competition and a whole host of other symptoms.

Even the most casual inspection of the consumer electronics market shows that this is not the case. One might at first suspect this is unusually virtuous behavior on the part of JVC; while it is true in my experience that they are a fine firm, this is not a complete answer. Probing further, one could ask how many standards exist for music CDs? The answer of course is only one - a consortium composed of Sony and Philips licenses 100% of that market, again without the deleterious effects one would expect from such a complete monopoly. Indirect competition from other prerecorded music formats might at first blush seem to be a factor, but it is not a sufficient explanation.

A new breed of economists, such as W. Brian Arthur [REFERENCE] of Stanford University have - realized what a small number of companies have known for some time - that there exist some fundamental exceptions to classical economic theory which require new insights and laws to explain them. The classical notions of monopoly simply do not apply.

The phenomena responsible for the VHS triumph over Beta is a positive feedback cycle. Consider the situation early in the VCR battle when VHS started getting more popular. As the players proliferated, video rental stores tended to stock more VHS tapes than Beta - the cost of having both in equal numbers was too onerous. The owner of a VHS player was therefore more likely to find the movie he wanted at the video store than a Beta owner. This made VHS fundamentally more useful to its owners, and caused even more people to buy VHS players, which in turn further incented video stores to stock VHS. Positive feedback existed for VHS market share.

Mathematically speaking, any process of this sort is fundamentally unstable. Perfectly balanced competitors could maintain their market share, but once an imbalance started it would grow without bound. The initial lead that allows one contenders to pull away from the others might be the result of a deliberate action or it could even be random chance. In either event, the initial lead will grow at an exponential rate until it captures the bulk of the market.

In the case of VHS versus Beta, there were several factors that caused VHS to pull away - VHS tapes could record two hours of video versus one hour for Beta - a critical difference if you are going to record a movie or football game. VHS was licensed to many manufacturers which produced lots of competitive models, whereas Beta was proprietary to Sony. Regardless of these factors however, the laws of positive feedback economics dictate that there could be only one winner. The existence of prerecorded video tapes and similar factors provided the mechanism for positive feedback which forced this convergence.

The computer industry as we know it today is full of examples of positive feedback. The value of a computer to its user depends on the quality and variety of the application software available for it. The incentive to create such software for a particular computer depends on the number of users, since they are the potential customers for the application developer. This creates a similar situation to the video store - the best software is attracted to the most popular platform, making it more popular still.

This is not the only source of positive feedback however. If I want to exchange data with you, or get advice from you, then it helps a lot if we are using the same computer and the same software. When a user upgrades to a new version there is a large benefit if existing data files can be used directly, thus favoring whatever software the user had in the past. A user who invests time learning the interface and commands of a piece of software will be loathe to re learn for a new package unless absolutely necessary.

A more familiar way to say this is *compatibility* - the laws of positive feedback govern any system where compatibility with other users is either directly or indirectly a key factor in the utility of a product or service. This value is usually instantiated and made tangible by a separate product whose availability or quality depends on the installed base, such as the rental tapes in the case of VHS or software in the case of computers.

The positive feedback effect has been responsible for the phenomenal strength of leading software products in both applications and operating systems. Digital Research, Novell, Apple and Microsoft are among those who have been beneficiaries of this effect at various times and in various market segments for systems software. Micropro International (developer of WordStar), VisiCorp and Software Arts (marketer and developer of VisiCalc), Lotus Development, WordPerfect, AutoDesk (developer of

AutoCad), Ashton-Tate (developer of dBase) and Microsoft are examples of companies who have been high market share leaders of specific applications categories at one point or another.

The historical situation is that the market share leader in systems software takes about 90% or so of the market, the runner up takes about 90% of what is left and so on. This maps reasonably well to the current world wide market share figures - MS Dos based computers have about 91%, Apple Macintosh computers which run the Macintosh OS have about 8% and the largest variant of UNIX has less than 1% [GET PUBLISHED FIGURES]. Applications software usually gets a somewhat smaller benefit from positive feedback than systems software because the effects which drive the feedback are less central to how someone uses the application. As a result the typical figures are something like 60% to 70% share for the leader, 60% to 70% of the remainder for the runner up and so forth.

As Arthur points out [REFERENCE], the positive feedback effect is also responsible for a variety of standard conventions, such as the definition of "clockwise" rotation for the hands of a clock, or which side of the road a car drives on. Unlike these rather static domains, technology based standards are inherently perishable. The pace of technical development soon renders any given piece of technology obsolete. The current king of the market share mountain created by positive feedback effects may have his day in the sun, but that victory can be fleeting. In fact, many of the companies listed above have had exactly that fate occur to them - they either no longer exist, or if the do are but a shadow of their former selves. The same fate awaits VHS, because within the next several years it is clear that a new digital tape format will arise and render any analog videotape format obsolete.

Any entrant in a technologically advancing market must reinvent itself and its product line on a continual basis, because any individual product will be obsolete - even to current customers - in a short period of time. Depending on the legal status and complexity of your intellectual property, competitors may also have the option of legally *cloning* your technology so that in the key area of compatibility, they are just as good as you are. The rate at which a software asset can be cloned is usually even faster than the rate at which it becomes obsolete. The only antidote to cloning is to stay one step ahead by continually extending your product with compelling features and doing so more rapidly than they can be duplicated.

As a general rule of thumb, a product in the computer industry - whether hardware or software - is only as strong as its current version. A single strong release, or a weak one can make or break a product, or company. In most cases a single version is marketed for between one and two years until it is replaced, so there is little time to rest on ones laurels. The strongest products might have a bit more leeway, but despite all of the advantages that accrue to the incumbent leader, no product could expect to survive two consecutive bad versions - at least if its competition is awake. Conversely it takes two consecutive good versions (and thus 3-4 years) to establish a newcomer. When a product does fall behind, the positive feedback cycle becomes a double edged sword because it will help the challenger just as surely as it helped the leader in a previous round.

In addition to technical obsolescence and outpacing the clones, achieving superior distribution and third party support are critical factors. The benefit of an operating system to a customer depends on the number of application packages for it, and the number and variety of computers that can run it. If you adopt a restrictive policy with applications developers so that the system is not "open", then you are creating an opportunity for a competitor to get ahead of you by being more solicitous to those developers. If you restrict the licensing of the software to certain manufacturers you leave the same gaping hole. Since there is plenty of benefit to be had in being a market share leader even if you are quite open, why be greedy? The result is that most companies that have figured out the power of positive feedback cycles adopt very open and reasonable licensing policies. It would be short sighted and counterproductive to do otherwise.

These factors explain why high market shares created by positive feedback lack the negative symptoms of a traditional monopoly. The market share leader must maintain an extremely competitive posture with respect to pricing, technological innovation and openness, or else he risks a long and irreversible fall from grace. In a sense, the leader is a prisoner both of his own success, and the process which put him there in the first place.

MS Dos, a product of my company, is a good example. MS Dos is by far the most popular operating system in the world, with more than 120 million users. Computer manufacturers license MS Dos for price which generally works out to be about 1% of the price of the system, and do so without discrimination. This is an incredibly small price compared to the value that it brings to the customer. In a typical PC it is by far the cheapest major "component" - the CPU chip, memory chips, monitor, power supply and even the sheet metal case all cost the manufacturer more. As it happens, this is in the same general ballpark as the licensing fees for patents or hardware standards, including VHS and CD. MS Dos is also cheap by comparison with the software sold to run on top of it. The world sells about $20 dollars [CHECK THIS!] of MS Dos compatible application software for each dollar that Microsoft makes from MS Dos itself.

In addition to being cheap, MS Dos is also the quintessential example of open software - nearly any company can license it (over 2000 manufacturers in XXX countries do so at present[CHECK THIS]) , and anyone can develop application software for it, with hardly any financial barrier to entry (just the cost of a PC and inexpensive software development tools), and without any permission or notification due to Microsoft or anyone else. MS Dos has been continually improved since its introduction over 12 years ago, and is currently at version number 6 (with version 7 in active development).

The laws of positive feedback tell us that if it wasn't Microsoft with MS Dos, then some other company with some roughly similar operating system would be in precisely the same position. All current operating system technologies place a strong value on compatibility, and thus provide the hook for positive feedback. The inexorable pull of positive feedback cannot allow a situation where there are several equal competitors sharing the limelight - except in the brief moment when a newcomer on the way to the top passes the former champion on the way down.

If MS-Dos had failed, in all likelihood the market share leader would be CP/M-86 from Digital Research. Their CP/M-80 operating system was the market share leader for the previous generation of personal computers based on the Z-80 and 8080 chips. Although few people remember this, IBM actually shipped *three* operating system choices for the original IBM PC - MS-Dos, CP/M-86 and the UCSD Pascal P-system. MS Dos won out in a contest which was in many ways more evenly balanced than VHS versus Beta. The critical factors included a few important technical details (CP/M 86 could not handle programs larger that 64K bytes in early versions) and superior execution.

Software Leverage

Suppose that you were to write a piece of software for your own use. This is typically the case for users of mainframe and scientific computers. In this case, for every dollar spent on development (programmer salaries etc.) you get a dollar's worth of software - if you do a good job. There is no leverage in the financial aspect of the development process.

Somebody else who gets their software from a small software company or value added reseller, as is typical for minicomputers and workstations, will find significant leverage versus their own development. The company they license the software from will have somewhere between 1,000 and 10,000 customers, and in order to stay competitive will wind up spending 10% to 20% of revenues on development. The leverage is now between 100 to 1 and 1000 to 1 - you are buying software at a tiny fraction of the development cost. Some of those costs may go toward features which are not of direct value to every customer, but on balance the ratios are so extreme that it is still a very significant effect.

In the personal computer world, the normal source of software is off the retail shelf. The market is so large that the financial leverage in development is astounding. You can go to the local software store and spent $100 to get software that cost my company $100 million dollars in direct development costs - primarily programmer salaries - to create. This isn't unique to Microsoft - the same sort of 1 million to one leverage is found in the products of other companies as well. It is an incredible situation for the customer - the leverage is so extreme that you can get features and support which would be unthinkable if you were writing the software for yourself. This is why MS Dos got to be so popular. The system was first shipped in 1981, and due to the technology extant at that time it is a variety of limitations.

Nevertheless, it has remained popular for many years because it became the platform for the best (and most highly leveraged) application software in the world.

The difference in software leverage is, in an economic sense, the real distinguishing feature between "PCs" and "workstations", "minicomputers" and "mainframes". The size of each market, and the concomitant difference in software leverage is more important than the technical details. If a customer's needs can be satisfied by software available for PCs, then they will be getting better software for less money - by a huge multiple - than in the smaller worlds.

Technology only enters in when it makes it impossible to serve those needs within a certain base. As we shall see below, advances in microprocessors will erase most technical barriers which up until now have protected the low leverage world of mainframes, minis and workstations from the high leverage world of personal computers. In another domain, the ability to create application programs with a graphical user interface is fundamentally lacking in MS Dos, which created an opportunity for a new graphical system to win.

Software leverage is simply a way to quantify the positive feedback cycle discussed above. Not only do the leading products get high share, but they also get a high absolute installed base because they can deliver so much more value to the customer.

The Architecture Win

Since compatibility is such an important feature, it is useful to create a term that describes the aspects of a high level design which capture the essence of compatibility without the specific details that may change from one version to another. The computer industry uses (and sometimes abuses) the word *architecture* for that purpose. In a world where the details of technology become obsolete at a rapid pace, it is very important to have a strong architecture asset. Architecture is the focal point of the positive feedback cycle.

When a customer selects an architecture by buying a product which instantiates it, the supplier has achieved an "architecture win" with that customer. The architecture win is potentially a long term commitment with serious consequences. There will have a strong incentive for the customer to remain with the architecture in the future, or a clone of it. The strength of this incentive depends a lot on the specific product - the more basic or fundamental the product, the more pain will be involved in switching. Switching to an entirely different computer or operating system (such as from a PC to a Macintosh) entails changing all of your software, whereas switching your word processing software has more limited consequences.

It is always possible to switch architectures at some point down the road if sufficiently motivated, for example if the architecture supplier screw up on two consecutive versions, or fails to maintain technical leadership. However, in the event that the supplier does keep up, a customer will generally stay with him, so the architecture win is a very valuable asset.

Given this, the question arises as to how a company should profit from the architecture wins that it achieves? What business model does one apply to the architecture win?

First generation computer companies, such as IBM, DEC and others in the mainframe or minicomputer markets understood the concept of architecture win, but they did *not* understand the positive feedback cycle and their business model reflected that. Their strategy was to use the architecture win to get high margins on all sales to their customer base. If you started as an IBM customer, then you would have to buy all software, accessories, service and any future compatible systems from IBM, with the architecture win margin tacked on.

This was an incredibly lucrative business model if you focus on the trees rather than the forest, because the value in the architecture win was extracted as a multiplicative factor on all hardware and software. If you took a purchase order for $1,000,000 worth of mainframe or minicomputer and accessories from the heyday of this business model and rewrote it so that "architecture win" was an explicit line item with normal manufacturing profits on all other items, then you'd find that it might be $250,000 or more.

This proprietary approach was enforced with unique features in the hardware and system software architecture which made it incompatible with other manufacturer's products - sometimes aggressively so, in order to prevent "plug compatible" or clone products. Unfortunately, the same barriers also produce a highly fragmented market for any products, such as software, which could instantiate the value of compatibility and get the positive feedback cycle going. As a result, the positive feedback either did occur, or its effects were limited.

The IBM PC represented a major innovation in the business model because the key elements of its architecture were available from third parties - principally Intel and Microsoft - so that other manufacturers could offer compatible machines. The effect was that the architecture win was sold directly, for a low fixed price rather than being a multiplier on the total hardware and software bill.

This decoupling of architecture win resulted in far lower revenue. A hypothetical $1,000,000 purchase order for personal computers would have on the order of $2,000 to $5,000 in the architecture win line item, including both hardware and system software architecture win fees. This revenue drop might have been disastrous had it not been for the compensating feature that the decoupling also enabled the positive feedback cycle to begin. The resulting explosion in growth created the personal computer industry as we know it today, with 120 million machines worldwide, running software leverage ranging up to one million to one.

The business model which decouples the architecture win from the total system is sometimes referred to as "commoditizing" the system. This is anathema to the old style proprietary thinking - to the point that it is widely regarded as being antithetical with having a healthy company or industry. The label "commodity" is really a misnomer since it implies products which have a low level of innovation and are all alike - in fact the opposite has been true.

There are two reasons for this. First, the positive feedback cycle drove unit volumes and therefore the absolute profit levels upwards in an exponential spiral. Activities such as research and development of new architecture features do not need enormous margins to fund them - they just need certain absolute levels of funding.

Second, the decoupling of the architecture - i.e. compatibility - from hardware to software freed designers to create and innovate without bearing the stigma of incompatibility. The PC market today is actually composed of a far wider variety of hardware than at any time in the past five years. New graphics adapters, hard disk arrays, peripherals and faster processors crowd the pages of PC magazines. Product reviews typically show performance and other key parameters varying by a factor of two or more between competitive products - hardly an example of indistinguishable "commodities".

Folks like Eckhard Pffifer of Compaq, Michael Dell of Dell Computer and Ted Waite of Gateway have found that it is possible to build efficient and thriving businesses in this environment. Their margins are reasonable for a manufacturing business - they only seem low when compared to the old style proprietary margins, however the increase in unit volumes more than makes up for this. The level of innovation must remain very high - nobody can afford to rest on their laurels for long, and there is a great deal of competition. Efficiency is at a premium; slow moving companies with fat cost structures need not apply, but many firms have found that PC hardware is a great place to be.

In a striking reversal to so many other industries, most of these products, including many of the very cheapest, are manufactured in the United States. In fact, US companies like Compaq and Dell have gone so far as to export a PC price war to Japan, forcing an unprecedented restructuring of that market. The Japanese approach of steady incremental improvement of the manufacturing process which has served so well in many other products does not work here. The pace of innovation and technical revolution must be very brisk, and to date this has been very difficult for offshore manufacturers to match.

How has this effected IBM, which started this process? Some authors, such as Charles Morris and Charles Ferguson [REFERENCE] have argued that working with Intel and Microsoft was a "mistake" on IBM's part, and that they should have used their own internal operating system and chip technology in order to maintain control of the architecture. In my opinion this is dead wrong. Maintaining control

runs counter to the openness required to get the positive feedback cycle. It is extremely doubtful that IBM, which was a master of the old proprietary architecture business model could have made the right moves to ignite the PC market.

In fact, IBM actually did create a number of personal computer projects with varying degrees of internal technology in addition to the one which became famous. All died on the vine. So did similar experiments with proprietary PCs made by Xerox, DEC, Convergent Technologies, Hewlett Packard, Texas Instruments and a host of smaller Silicon Valley startup companies like Mindset, Fortune Systems and others. Had the IBM PC been based on IBM processor and operating system technology, it never would have amounted to much more than a historical footnote.

Perhaps the biggest flaw in Ferguson and Morris' argument however is the fact that the open PC market was actually a boon to IBM. The PC put IBM at the head of one of the fastest growing and most dynamic phenomena in the annuals of business, and extended its hegemony to a new market in a manner which none of the other mainframe or minicomputer companies have been able to match. The PC business remains one of the strongest parts of the IBM today - the company's financial woes are primarily due to mainframe divisions that solidly followed the old proprietary architecture business model.

In fact, to the degree that IBM has had problems in the PC market, it invariably came as a result of too much proprietary thinking rather than too little. Slowly but surely they squandered much of the leadership asset that had accrued to them - mostly by following the sort of advice that Ferguson and Morris give them ex post facto.

If you have the old school approach to architecture win, then the operating system appears to be an ideal way to reinforce your architecture win and raise the margin on the entire computer system. IBM was forever trying to use the operating system to do this. The most grievous example was OS/2, a project that I worked on early in my career at Microsoft.

The original intention of the project was to take Microsoft Windows and host it on the new operating system kernel that had been jointly created by IBM and Microsoft as a successor to MS Dos. Alas, the temptation to use IBM unique technology took hold. The first decision was to make the graphics portion of the system compatible with an IBM mainframe package called GDDM. This broke compatibility with Windows, and started the product down the path of ruin. GDDM at the time had about ten thousand customer licenses - a great success by mainframe software standards. The notion of bending a product to gain advantage in a market serving ten thousand customers at the cost of disenfranchising the PC industry and threatening the program's competitiveness was considered a sound tradeoff by senior IBM management. It met their strategy of using IBM technology to control the architecture. Arguments to the contrary were made both by Microsoft, and many inside of IBM, but to no avail.

If GDDM compatibility had been the end of it, the product might have survived, but soon afterwards a program called Cross System Consistency, later renamed the Systems Application Architecture or SAA took hold. SAA was designed to be a grand software architecture that would bolster IBM's hardware from PCs to mainframes. OS/2 was critical to this strategy, because it would deliver the PC market momentum behind SAA and therefore behind a variety of IBM mainframe standards. OS/2 had to be changed to make it conform to these standards. The alterations came in to the joint IBM/Microsoft design team as Design Change Requests, or DCRs - little forms filled in by people in any IBM product group who wanted to alter OS/2. The DCRs soon became a blizzard - the first graphical version of OS/2 suffered over *twelve thousand* of these DCRs [CHECK THIS], churning the design of the product in nearly every way.

The final result was a product so divorced from the market realities of the PC world that it was unacceptable in the very market it was supposed to have been designed for. Despite our best efforts, IBM would not admit failure and back off - the strategic control that they thought it might give them in hardware margins was too alluring to give up. In the end, OS/2 finished off most of IBM's reputation as a software company, and their refusal to work with Microsoft on Windows 3.0 cost them the ability to show leadership in the most dramatic paradigm shift that occurred since the original IBM PC loss.

Independent estimates [Paul Carroll's book??] put the total IBM financial loss for OS/2, and OS/2 specific software projects that were created and canceled along the way, at over $500 million [CHECK THIS]. If you compare compared to what they could have had the total opportunity cost is in the billions.

Scalability

Although the old style computer company didn't understand positive feedback, they did capture a portion of its power with the notion of architectural scalability. In fact, the discovery of the principle was the primary mover behind both IBM and DEC.

In the mid 1960s, the mainframe computer market was highly fragmented. Each new model of a machine - even from a single manufacturer - would be different from the next, with incompatible hardware and operating systems. The machines were also quite specialized - some were "business" computers and other were "scientific". There were plenty of technological rationalizations for these specializations, and it was widely believed that they were necessary. Every time you switched machines - whether to move up to a more powerful one, or replace an old one, there was a painful conversion process for old software.

IBM made the bold move to change this with the introduction of the System 360, which was designed from the beginning to be a scalable, general purpose architecture. In fact, IBM coined the term architecture to describe the (then) radical concept of having multiple machines at different price points and performance levels which were mutually compatible. The actual internal design and implementation of the various members of the System 360 family was totally different, as dictated by their chosen price and performance targets, but the had the same architecture - and thus were compatible.

Rather than design one machine and then try to achieve "backwards compatibility" by cloning it, IBM planned up front for a design that could be built at many different sizes and scales. Scaling over performance levels also allowed them to scale over time, because as technology advanced what used to be a high end lever of performance would shift down the price curve. They also unified the difference between business and scientific machines, by making System 360 support both.

The result was a huge advantage to both customers and IBM. Customer problems came in all sizes, and now the same computing system could too, without throwing away software and starting all over again. The runaway success of System 360 made IBM the powerhouse in mainframe computers, which it successfully held for the next thirty years.

A decade later, DEC found itself in a similar position. The minicomputer market was just as fragmented between different systems as the mainframe market had been for IBM. DEC's first truly scalable architecture was the VAX, and its impact on the minicomputer market was similar to what System 360 did in mainframes. DEC produced dozens of variations of the VAX, ranging from a MicroVAX implemented on a single chip to the VAX 9000 which used several circuit boards of exotic ECL chips to implement the CPU.

Scalability is, in some sense, a simple approximation to the positive feedback cycle. The fundamental benefit is the same - compatibility becomes valuable to customers, so the value of the architecture grows in proportion to the total installed base. The degree of leverage and feedback obtained still fell far short of what the personal computer did many years later, but it was far higher than what could have been achieved by a product line of separate and incompatible products.

These days, the lesson of scalability is de rigeur for hardware designers. Every major CPU architecture spends a tremendous amount of energy ensuring that they can scale across a wide range of price and performance levels. The various models of Intel processors found in most personal computers can scale over a performance range of nearly 100X. Nobody would propose a non scalable CPU.

This is not the case for operating system software. In fact, the situation is actually very similar to the degree of fragmentation that existed in the mainframe market prior to System 360, or the minicomputer market prior to VAX. There are dozens of different operating systems available today - most of which

are incompatible with each other. The systems are also quite specialized to niche markets, analogous to the split between scientific and business computers. One operating system is "pen centric" - based on the notion that you need an entirely different operating system to run on a machine which uses handwriting recognition rather than a keyboard. Another is a "network operating system" designed exclusively to be a server. Other niches include workstations (usually a proprietary variant of UNIX), PDAs, handheld pocket computers, smart TVs and others. Each of the developers of these systems has a rationale for why it is better to specialize.

In addition, current operating systems have not been designed for multiple implementations at different scales. Instead, people generally "port" an operating system when they want to run it on a different machine. This consists of making minor adaptations to run the same code. The analogy with chips would be to have had all VAXes, from the MicroVAX to the VAX 9000 use the same CPU chip. This would never work of course - the design appropriate for the small machine would not have the horsepower for the advanced one.

I believe that scalablity will sweep the world of operating systems, and be just as powerful there as it was for hardware. This concept is a fundamental part of the Windows Scalable Architecture which Microsoft is currently deploying. Around four years ago we created an architectural specification which was designed to allow multiple software implementations. Mainstream office PCs with Intel processors use the implementation know as Windows 3.1. Higher end desktop machines using either Intel processors or RISC based processors such as MIPS will use an implementation which was completely rewritten, known as Windows NT 3.1. Server machines, including multiprocessors, will use the Windows NT Advanced Server version. Meanwhile, at the low end we are working on home and consumer versions of Windows. The Microsoft AtWork architecture extends key elements of Scalable Windows into intelligent office appliances - FAX machines, copiers, smart telephones and pocket computers.

One way to view the scalable operating system strategy is as a mechanism for exploiting software leverage in markets other than PCs. Today, the workstation, minicomputer and mainframe markets are the dominion of low leverage software, because there are many incompatible systems fragmenting the market. A scalable operating system which covers both the mainstream desktop PC and the realm of high end desktops and servers will also extend high leverage software to those areas. Software developed for smaller machines can migrate upward. Even more important, people can develop new classes of applications for servers. The personal computer is not really a small mainframe - the combined innovation of software developers exploiting the inherent leverage in that market created many new programs which revolutionized the work of individuals. Server software has the same potential for work groups and organizations, and I believe that the emergence of scalable operating systems (driven by positive feedback) will enable this revolution to occur.

The move to a scalable architecture for system software is controversial, just as it was for hardware at one point. Only time will tell whether or not the analogy is complete and this makes sense or not, but the potential certainly seems to be there.

The Indentured Software Company

Not all of the old school proprietary companies started in the mainframe era - many are the product of Silicon Valley. Apple Computer and Sun Microsystems are too good examples - their fundamental business strategy is virtually identical to the old style computer companies. The only real distinction is that the vast majority of the architecture win they merchandise is contained in their system software rather than in hardware. In fact, senior executives of each company have frequently been quoted making statements like "We are really a software company that happens to sell hardware". [REFERENCES]

The basic proposition is simple - write great software but only sell it bundled with your own hardware, with a high margin on the total package. The hardware may have some special features, but unlike mainframes and minis it is built like a PC - using widely available processors and other components. Rather than being a software company that sells hardware they are really a company following an old style hardware business model, with a captive software company providing the value. If you take a

software centric point of view, then the end user price of the Apple architecture win - principally the operating system - is perhaps 10% to 20% of the system cost - or about 10 times greater than is typical for a pure software company business model.

Apple has a very competent software group, and they have several software joint ventures such as Kalieda and Taligent, but the software business aspirations of these groups will continue to be held hostage to the hardware oriented business model unless Apple changes that model entirely.

The Apple Powerbook is a fine example of the challenges that come up in reconciling hardware and software business models. The Powerbook series is one of the great success stories of the computer industry, achieving about $1 billion in sales in its first year. Apple should be justly proud of this accomplishment, but there is a dark side to story. Toshiba, Compaq, NEC, Grid and a variety of others had pioneered the laptop market with IBM PC compatible machines nearly five years before Powerbook.

How many billions of dollars in revenue did Apple leave on the table by coming in five years late? Presumably the hardware division had other priorities which got in the way of them making their own product, but from a software perspective this was also a failure in the business model. If instead Apple had licensed its system software to others there would have been Mac compatible laptops coincident with those for the PC. The real tragedy for Apple as a software company is the loss of many users who were forced out of its architectural camp and into that of the competitors because there was no laptop solution.

Sun Microsystems' emphasis on software has been higher even than Apple's - to the point that they organized their system software business as a separate company, SunSoft. Despite this, SunSoft has had a hard time being a really independent software company. The original rhetoric behind SunSoft is that it would service a market of machines which were hardware compatible with Sun's own hardware, powered by the SPARC chip which Sun developed and openly licensed.

If this had really been achieved, Sun would have actually made the difficult leap toward having a software centric business model. Their hardware division would have to compete on the same sort of basis that Dell, Gateway and Compaq do these days, because they could no longer count on the architecture win to . Unfortunately, the SPARC compatible market never quite got off the ground. The usual reason cited is that Sun could never bring itself to license enough intellectual property (both hardware and software) for people to make a competitive machine on a level playing field. This may have been the best decision for the company as a whole, because Sun's overall business model is that of a hardware company.

Sun is but the most successful of a long line of people with proprietary operating systems based loosely on the UNIX system. Originally developed by Bell Labs, UNIX was widely licensed to computer companies and adapted for their various machines. The initial attraction was that a small company which did not have the resources to develop an entire operating system could simply port UNIX and suddenly have an OS. Unfortunately, none of them seems to have been able to resist the temptation to also introduce their own proprietary features, so that rather than being a standard, UNIX became a proprietary OS construction kit. Like Sun, the others in the UNIX camp never managed to ignite the positive feedback cycle, and

All software development efforts managed run by a company with a business model oriented toward proprietary hardware runs into this problem - how do you make the tough decisions which favor broad software and positive feedback cycles versus keeping software indentured to your own hardware?

Becoming a software company is hard even if you bite the bullet and simply quit the hardware component of your business, as NeXT has recently decided to do. The kind of software that you do as a hardware company is different in terms of its technical strategy than what you'd do as a independent software company. As a hardware company, NeXT needed an operating system, so their software group assembled one from a combination of internal development and components licensed from outside. The core of what NeXT created was an object oriented development environment, which was well integrated with the rest of their operating system.

Now that NeXT is a software company, they have a difficult choice ahead of them. Their natural competitors in creating object oriented development tools include companies like ParcPlace Systems, Borland and Microsoft (with Visual C++ and other development tools). None of these packages includes an operating system - instead they run on top of other standard operating systems. It is hard for NeXT to compete with them directly because its customers would have to throw out the operating system (and potentially all the other software that depends on it) just to get a good development tool.

NeXT could fix this by porting the relevant parts of their system to other operating systems, but this will cost valuable time and lose some of benefits of their old system. Or, they could decide to compete with Apple, IBM, Taligent, Microsoft (with Windows NT and future systems) and the legions of UNIX developers. The necessary investment to compete with these companies is enormous, and even if they did, it would take the NeXT software developers far afield from the central value added that they created in the first place. Any way they decide to go, it is clear that the decision to integrate closely with the operating system which made sense as a hardware company now is a liability.

De Facto versus De Juris

One alternative to making the architecture win a valuable business asset is to place it in the public domain. This is usually achieved through standards committees, such as those operated by ANSI, ISO and CCITT. These "official" standards could in principle replace the de facto standards which are created by the positive feedback cycle.

Although it may sound altruistic to have a public standard, the actual record of such standards in the computer industry is rather poor. The nature of the committees that create them is to move only as fast as the slowest member, because nobody wants to be left behind. The result is usually mediocre - slightly worse that when the lowest achieving company on the committee could achieve alone. In essentially every case where a de facto standard driven by positive feedback has come up against a de juris committee standard, positive feedback wins. Paradoxically, the customer wins too - as recounted above, standards driven by positive feedback are typically very cheap and the investment in improving them (do to the effects of leverage) is enormously more than anybody could afford to spend on a public standard. There are few things more costly than a "free" standard.

In the telecommunications world such standards have been more successful, in part because the committee members were all regulated monopolies and had little ability to drive a standard commercially. In some cases, such as the Group 3 and Group 4 standards for FAX machines set by the CCITT, they have been responsible for rapid growth in a classic positive feedback cycles. The value of a FAX machine clearly depends on the number of people you know who also have FAX machines, providing the compatibility hook which allows positive feedback to work.

FAX is also a good example of the downsides of a de juris standard. The CCITT adopts a new FAX standard once every four years - an eternity considering that the fundamental technology driver for FAX machines is computing technology. As a result, nearly any FAX machine has a whole host of features which only work with a machine from the same manufacturer. The likelihood of a user being able to take advantage of these features is nil, but what else can a FAX maker do? The FAX standard enforces a commodity status for FAX machines - and unlike the situation with PCs it is a static and non-extensible standard.

Four years ago, my group decided to solve this problem with what was later named the Microsoft AtWork architecture - operating system software for FAX machines, PBXs, copiers, telephones and other business communications equipment. Many people initially thought that we were crazy to attempt to extend existing standards like FAX because it had been a part of the landscape for so many years.

Instead, we found that the concept was embraced by manufacturers - including Xerox, Ricoh, Murata, Canon, Hewlett Packard and sixty others so far. The reason was simple - AtWork allows an aggressive manufacturer to continually extend the feature set and functionality, yet still maintain compatibility. New features and functions can be added in both hardware and software, and across product lines such as FAX

and copiers. It is too early to judge whether positive feedback will kick in and make AtWork a success with users, but it is clear that the concept of an extensible system has great appeal in the industry.

Grist for the Mill

Much of this article is focused on the growth of market share that can result from the laws of positive feedback. This is a very powerful effect, but one cannot understand the computer industry without also factoring in the incredible trends in semiconductors and chip design. The growth in hardware capabilities provides the basic impetus to evolve software products. Greater computer power enables new features and capabilities. Increased data storage capacity gives programs more room to store information. Lower cost allows computers to appeal to new markets. Increases in all of these fuel the growth of the software industry.

The usual figure of merit in computing is the ratio of performance to price. You can keep the amount of computing power constant and watch the price decline, or keep the price constant and watch the total computing power increase - either way it effects this ratio. In the last 20 years this ratio has increased by a factor of about 1 million. Numbers like this are hard to fathom by themselves, so consider this example - a factor of 1 million speed up would reduce a computing task that would take a full year just over 30 seconds. All current indications are that we can expect at least another 20 years at the same pace, and thus *another* factor of 1 million in the price-performance ratio. Indeed many would claim that the rate of innovation has lately been on the increase.

This kind of growth in capability is completely unpredicented in other industries. Many other electronic items have a steep price decline, but the increases in computing power outstrip even these rates. The changes wrought by this kind of change profoundly alter the competitive landscape. You might think that computer companies have come to grips with this, but this is like saying that downhill ski racers are accustomed to speed. Although it is mostly true, many participants in either domain routinely fail because it is so easy to misjudge the effects, and so hard to recover once you do.

In addition to computing performance, storage capacity is also increasing dramatically. A typical PC today has both semiconductor memory, or RAM and secondary storage which is usually a hard disk. A typical PC today will have 4 megabytes of RAM, and a hard disk of about 160 megabytes. Projecting forward, we can estimate that a PC of comparable cost in the year 2000 is likely to have about 1 gigabyte of RAM, and a 40 gigabyte hard disk.

To put this in perspective, consider that American Airline's SABRE airline reservation system has a total data storage capacity today of about 40 gigabytes [CHECK THIS] - the same size that I'm projecting would fit on *one* desktop PC. Of course, SABRE might be larger by then - but probably not much larger. There are only so many airplanes and travel agents in the world and they only type so fast, and the SABRE database size depends on these factors. None of them are growing fast enough; even rabbits would have a hard time breeding fast enough to keep up. Most similar traditional data processing task will undergo profound restructuring because they grows at a rate constrained by factors in the real world. The exponential explosion in computing capabilities will overwhelm them.

What will this power be used for? Richer data types is a partial answer. A typical novel.or movie script is a megabyte or so as text, but a feature film made from it might be more like 3 gigabytes - a factor of 3000 greater. Video and audio will become commonplace in the computers of the near future, but even so these data types will only consume a piece of the available resources - a factor of 3000 will be completely absorbed in only a dozen years.

User interface issues have always been important, and this certainly will consume an increasing amount of CPU cycles. Another important use of this power will be to feed ever more extensive computations and analysis. One of the few things that grows at the same rate of computer capability is computing itself. Most data will not be for direct human consumption - it will be passed to software for further analysis. Today's spreadsheet user does "what if" calculations involving very simple financial models. Future decision makers will be able to simulate the impact of what they do, even for extremely complex systems.

Every discussion of future computing prompts some folks to ask why this power will be needed. Don't we have enough already? The answer of history is clear - none of these predictions has ever been even remotely correct. Software innovation has always been able to make up the gap, and there is no reason to believe that this will stop - for the next twenty years at least. As this occurs, the industry itself is reshaped by the laws of power feedback and the business models and strategies by which companies either harness them, or ignore them to their eventual peril.