Software Business Entrepreneurship
Lessons from Bill Gates and Microsoft
SWP 4046
Michael A. Cusumano
MIT Sloan School of Management
Fuji Conference, January, 1998
Revised Draft: December 10, 1998
Introduction

Established in 1975, Microsoft is now the largest company in the world specializing in computer software, with some 27,000 employees and annual revenues of exceeding $11 billion in 1997 (Table 1). Microsoft has grown by skillfully moving from one software mass market to another. The company started by selling programming languages for a new PC kit, and then built an operating system (MS-DOS) for IBM and then other computer equipment manufacturers. It gradually entered a variety of applications markets and sold products directly to retail stores. Today, Microsoft dominates various PC software markets throughout the world. It has a market share of around 90 percent in operating systems (Windows and Windows NT), and this has provided the company with an enormous revenue stream for many years. But Microsoft has also learned how to develop applications programs, and builds more than 200 software products. In addition, the company has an Internet portal business and is rapidly increasing its presence in the Internet software (browser and server) markets, where it entered late but now has a market share of approximately 60 percent in browsers (Cusumano and Yoffie, 1998).

In fact, Microsoft continues to grow at an astounding pace for a large firm (see Table 1). The company has also created incredible wealth: Microsoft’s market valuation in 1998 reached approximately $300 billion. Bill Gates is frequently listed as the richest individual in the world, with a net worth (mostly Microsoft stock) of more than $50 billion. Several thousand of Microsoft’s employees are also millionaires due to their generous stock options, and Gates’ generosity here has helped him retain the best employees.

Clearly, there should be much that high-tech entrepreneurs can learn from Bill Gates’s management style and Microsoft as an example of American-style entrepreneurship in the personal computer (PC) software business. Accordingly, this paper focuses on three different themes that characterize the success of Bill Gates and Microsoft as an entrepreneurial company.1

The first theme -- understanding both the technology and the business -- emphasizes the point that the software business is highly technical. Therefore, it is extremely difficult and usually unwise to separate technology decisions from business or strategic decisions. Microsoft has never had to struggle with this problem because Bill Gates grew up with computers, has been one of the pioneers of PC programming technology, and has displayed a remarkable ability to understand strategic thinking and make money off his technical skills.

The second theme -- building an organization of “smart people” -- emphasizes the point that developing products for the software industry is a highly creative and intellectual activity, and therefore successful computer software companies must be built around “smart” people. Not only does computer programming combines elements of art, science, and engineering, but the technology has been evolving so rapidly that managers and engineers must be able to make decisions very quickly and on their own. This also requires smart people. Gates has always understood this issue and has been extremely selective in recruiting people for his company. In addition, following the example of Bill Gates, the word “smart” has come to have a special meaning in Microsoft: the ability to learn and solve problems quickly, as well as the ability to understand software technology and how to make money with this knowledge.

The third theme -- balancing structure and flexibility -- emphasizes the point that a successful software entrepreneur and entrepreneurial company needs to have enough flexibility to allow creative software developers to create and invent products and new technologies as necessary. At the same time, however, a successful software company also has to have enough
structures and processes in place to coordinate and focus the work of these creative people. This requires a delicate balancing act in management, which Bill Gates and Microsoft have also done extremely well.

### Table 1: MICROSOFT COMPANY DATA, 1975-1997

<table>
<thead>
<tr>
<th>Year</th>
<th>Revenues ($1,000)</th>
<th>Growth Rate (%)</th>
<th>Employees (people)</th>
<th>Operating Profit (%)</th>
<th>R&amp;D/Revenues (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1975</td>
<td>16</td>
<td>--</td>
<td>3</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1976</td>
<td>22</td>
<td>275</td>
<td>7</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1977</td>
<td>382</td>
<td>1636</td>
<td>9</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1978</td>
<td>1,356</td>
<td>255</td>
<td>13</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1979</td>
<td>2,390</td>
<td>76</td>
<td>28</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1980</td>
<td>8,000</td>
<td>235</td>
<td>38</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1981</td>
<td>16,000</td>
<td>100</td>
<td>130</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1982</td>
<td>24,486</td>
<td>53</td>
<td>220</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1983</td>
<td>50,065</td>
<td>104</td>
<td>476</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1984</td>
<td>97,479</td>
<td>95</td>
<td>778</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1985</td>
<td>140,417</td>
<td>44</td>
<td>1,001</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>1986</td>
<td>197,514</td>
<td>41</td>
<td>1,442</td>
<td>31</td>
<td>11</td>
</tr>
<tr>
<td>1987</td>
<td>345,890</td>
<td>75</td>
<td>2,258</td>
<td>37</td>
<td>11</td>
</tr>
<tr>
<td>1988</td>
<td>590,827</td>
<td>71</td>
<td>2,793</td>
<td>32</td>
<td>12</td>
</tr>
<tr>
<td>1989</td>
<td>803,530</td>
<td>36</td>
<td>4,037</td>
<td>30</td>
<td>14</td>
</tr>
<tr>
<td>1990</td>
<td>1,183,000</td>
<td>47</td>
<td>5,635</td>
<td>33</td>
<td>15</td>
</tr>
<tr>
<td>1991</td>
<td>1,843,000</td>
<td>56</td>
<td>8,226</td>
<td>35</td>
<td>13</td>
</tr>
<tr>
<td>1992</td>
<td>2,759,000</td>
<td>50</td>
<td>11,542</td>
<td>36</td>
<td>13</td>
</tr>
<tr>
<td>1993</td>
<td>3,753,000</td>
<td>36</td>
<td>14,430</td>
<td>35</td>
<td>13</td>
</tr>
<tr>
<td>1994</td>
<td>4,649,000</td>
<td>24</td>
<td>15,257</td>
<td>37</td>
<td>13</td>
</tr>
<tr>
<td>1995</td>
<td>5,937,000</td>
<td>28</td>
<td>17,000</td>
<td>34</td>
<td>14</td>
</tr>
<tr>
<td>1996</td>
<td>8,671,000</td>
<td>46</td>
<td>20,000</td>
<td>35</td>
<td>17</td>
</tr>
<tr>
<td>1997</td>
<td>11,358,000</td>
<td>31</td>
<td>27,000</td>
<td>45</td>
<td>17</td>
</tr>
</tbody>
</table>

Note: "na" refers to "not available." Microsoft began publishing data on operating profits after going public in 1986.

Sources: The major sources for this table are Microsoft annual reports from 1986 to 1997, and, for prior years, Philip M. Rosenzweig, “Bill Gates and the Management of Microsoft” (Boston: Harvard Business School, Case #9-392-019, 10/9/92).
Understanding the Technology and the Business

Computer software has special characteristics as a technology and as a business. Understanding these characteristics is essential for an entrepreneur to succeed in the software business. Bill Gates has this understanding. He became familiar with computers and programming in junior high school. In fact, he was one of the pioneers of PC software technology and the first person to establish a dedicated PC software company. Gates also showed an early interest in making money, and had a software company in high school with partner Paul Allen, co-founder of Microsoft. We also know that Gates has been an avid reader of business history and biographies of great leaders in world history, and he continues to apply what he has learned in managing Microsoft. As a result, Gates has positioned himself to understand deeply both the technology and the business of computer software. He has also played several roles simultaneously in Microsoft. Gates has been both an entrepreneur (the company founder) and key technologist. He has also been the chief strategist as well as an effective leader and manager.

Software as a Technology and a Business: We can observe several unique characteristics of computer software as a technology and as a business. First, the technology consists of computer code -- instructions written in one of many programming languages -- running on computer hardware and over networks. There is no bending of metal or molding of plastic to create software products. But products can be large and extremely complex in terms of the number of components and their potential interactions. (Microsoft's Windows 95 operating system, for example, totaled about 11 million lines of code and took a team of about 400 developers and testers nearly 3 years to create.) This is truly a "knowledge" industry, where products consist of ideas and the key assets of a company are well-trained people -- often very young people. Product distribution is also increasingly "electronic" and "instantaneous" over telephone and cable-TV networks.

The people who write computer code are also unique. The popular press refers to them as "nerds" or "hackers" -- unkempt technical wizards who can make computer hardware do almost any task imaginable. Part of the fascination with software developers is that programming is more an "art" than a "science," though some companies try to manage software development like crude "engineering," using rules of thumb and historical data (Brooks, 1975; DeMarco and Lister, 1987). Entrepreneurs and managers in the software business need to understand the mentality and motivations of software programmers, as well as the peculiarities of the technology they create.

Another characteristic of software relates to what I call the mythical man-month syndrome. Managing software projects is notoriously difficult (there is something like a 10 or 20 to 1 difference in productivity between the best and worst programmers on a typical team). How long it will take to build a product is extremely difficult to predict because of the variance in individual performance as well as the uncertainty of the technology (programmers often have to "invent" functions as they go along). And people and months are not interchangeable; companies usually cannot speed up late projects by just adding more people, because the new people take time to learn what has been done and existing team member must stop and teach them what to do. Larger teams also create communications and coordination problems. Then testing software products -- finding and removing bugs -- is also incredibly difficult due to the huge number of
combinations of software and hardware products and user scenarios. Yet shipping at least “good enough” products in a timely fashion can be critical to success.

Not surprisingly, software programmers have created a special culture: The companies they prefer to establish and work in are notoriously non-bureaucratic, informal, and laid-back, but very hard-working. Work any 80 hours a week that you like, and the drinks are free, too! A hobby that is big business!

Then there is the rapid pace of change. Since the beginning of the software industry in the 1950s, the hardware technology and software programs have evolved with astounding speed. The industry is particularly fast-paced in certain segments where the hardware changes the fastest, such as PC software and Internet platforms and applications. This pace is due not only to the rapid evolution of hardware, driven mainly by Intel for personal computers, but also the pliability of the technology, and the ability of even individuals in garages to come out with significant product and technological innovations.

The pace of change means that the future is uncertain. Managers in the PC software business do not know precisely what their products and markets beyond a year or so in advance. Hardware evolution is a bit more predictable (it is guided more by the “laws” of physics). But how far and fast the software side will evolve, and when and how users will react to innovations, is usually guesswork. This means that entrepreneurs and managers in this business must be able to live with great uncertainty and constantly make educated guesses -- which is why having a deep understanding of the technology is often critical.

Another peculiar characteristic of software is the important role of technical standards and network externalities. Simply having a great product is not enough, in hardware or software. Software must work on computer hardware and systems software platforms, and different software programs must be able to work together. As a result, software companies have raced to establish a large enough presence to create a “standard,” or they have struggled to be “compatible” with existing standards created by others. What becomes the software standard also depends in large part on hardware standards and sales of hardware platforms. The term “network externality” comes into play here in that the value of a software product is often less dependent on the price or functionality of the product itself than it is on whether or not the product is compatible with the existing standard. This is “external” to the product itself. And the more computer systems that are sold which are compatible with the standard your product supports -- in other words, the greater the network of users for that standard -- then the more valuable your product and the standard become. Therefore, an essential part of being a successful software entrepreneur is to understand market dynamics and how to use customer and competitor behavior to your advantage. (This is something that Apple executives, from Steve Jobs to John Sculley, never learned. See Cringely, 1993.)

Finally, another characteristic of computer software is the tremendous opportunities the business presents to make money and have fun! The software industry has produced a remarkable number of millionaires and several billionaires (including Bill Gates, the world’s richest individual), even though most programmers say they are not motivated by money. And the market is only beginning to grow. There are 5 billion-plus people in the world, and perhaps 250 million users of PC and Mac personal computers. Furthermore, constant evolution of hardware capabilities means that software users, like it or not, are frequent repeat customers -- probably for life.
Gates the Technologist and Entrepreneur: When we look at the background of Bill Gates, we can see why he has done so well as a software business entrepreneur. He grew up with the technology, and has always shown skill in making money.

William Henry Gates was born in 1955 in Seattle, Washington, the middle child in a well-to-do family. Neither parent was a technologist; his father was a lawyer and his mother a school teacher. By all accounts, Gates the child was similar to Gates the adult. A former teacher described him as “a nerd before the term was invented.” His childhood interests included but did not end, obviously, with games such as Risk, where players competed for global domination.

Gates’ exposure to computers came during 1968-1969, in his second year at the private Lakeside School. The school had a primitive teletype machine and access to a computer through a time-sharing hook-up. He learned the BASIC programming language and then teamed up with a tenth-grade electronics expert named Paul Allen to learn more about programming. When Gates was 14, he and Allen formed a business writing and testing computer programs. The duo established their first company in 1972, named Traf-O-Data, and sold a small computer that recorded and analyzed motor vehicle traffic data.

In 1973, Gates enrolled in Harvard University, taking mathematics and computer science classes, among other subjects. The following year, Allen, who had gone on to study computer science at the University of Washington, left college and took a job with Honeywell in the Boston area. It has often been described how Allen saw an issue of Popular Electronics in 1974 that advertised the new Altair microcomputer kit from MITS Computer, and how he and Gates wrote a version of BASIC using Harvard’s computing facilities. Gates left college in 1975 to concentrate full-time on developing programming languages for the Altair and then for other personal computers. To do this properly, he and Allen (who left Honeywell) relocated to Albuquerque, New Mexico, to be next to MITS Computer’s office. They formed Microsoft during 1975 as a 60-40 partnership in favor of Gates, reflecting his larger role in developing Microsoft BASIC, the company’s first product.

Several characteristics stand out in stories about the young Gates. He was well above average in intelligence. He was ambitious. He had intelligent and ambitious friends. He was able to concentrate intensely and master what interested him. He became fascinated with computers and how to program them for practical purposes. He decided he wanted to make money at programming and quickly became an entrepreneur. Perhaps most important, Gates soon envisioned a world with computers not merely tracking traffic data but sitting on every desktop -- and running his software. This was a great combination of skills and ambitions to have at the dawning of the PC era.

Observers from outside and employees from within the company paint similar pictures of Gates. They describe him as a visionary with a maniacal drive to succeed, accumulate great power, and make money by taking advantage of his technical knowledge and understanding of industry dynamics. Microsoft the company emerges as an extension of Gates’ unique personality and skills:

“Gates is a visionary. Very early in the history of the PC, he evolved a strikingly clear concept of where the industry was headed, and he has pursued that vision -- despite many tactical setbacks -- unwaveringly, relentlessly, and ruthlessly.” (Charles Ferguson and Charles Morris in Computer Wars, 1993, p. 67.)

“This guy [Gates] is awesomely bright. But he's unique in a sense that he's the only really bright person I've ever met who was 100% bottom-line oriented -- how do you make a buck? Not in a mercenary way. It's just like that's what's good in life, right? If you make money, clearly that's good.
It's like he's this huge computing machine that knows how to make money.” (Interview with Jim Conner, Program Manager, Office Product Unit, Microsoft Corp., 8/2/93.)

“He's a maniac. Bill knows more about the product than any of us. And you go into the meetings and you come out just sweating because, if there is any flaw, he will land on it immediately and pick it to bits. He is just unbelievable.” (Interview with Dave Maritz, former Test Manager, Windows/MS-DOS, Microsoft Corp., 4/15/93.)

“IBM thought they had Gates by the balls. He’s just a hacker, they thought. A harmless nerd. What they actually had by the balls was an organism which has been bred for the accumulation of great power and maximum profit, the child of a lawyer, who knew the language of contracts, and who just ripped those IBM guys apart.” (Quoted by John Seabrook in The New Yorker, 1994, p. 59).

Perhaps even more impressive than his personal talent and knowledge of computers is the strategic insight that brought Gates to establish Microsoft as a pure software company. He claims, and his actions support the claim, that he realized early on that the real money in the computer industry was in software. This insight is remarkable for 1975. At the time, even Gates’ partner, Paul Allen, wanted to produce both hardware and software products. Most companies in the computer industry, led by IBM, DEC, and even new entrants such as Apple Computer (founded in 1976), also were focusing on hardware. Gates recalled his thinking in a 1994 interview: “I thought we should do only software. When you have the microprocessor doubling in power every two years, in a sense you can think of computer power as almost free. So you ask, why be in the business of making something that’s almost free? What is the scarce resource? What is it that limits being able to get value out of that infinite computing power? Software” (Playboy, 1994, p. 63).

**Gates the Strategist:** Since founding the company in 1975, Gates has directed Microsoft’s entry into a variety of related segments in the software industry. In doing so, Microsoft has followed a strategy that my book Microsoft Secrets described as pioneer and “orchestrate” evolving mass markets. Five specific principles characterized what Microsoft has done:

1. Enter evolving mass markets early with “good” products that set industry standards and sometimes generate other new markets.  
2. Incrementally improve new products and periodically make old products obsolete.  
3. Push volume sales and exclusive contracts to ensure that company products become and remain industry standards.  
4. Take advantage of being the standards provider with new products and product linkages.  
5. Integrate, extend, and simplify products to reach new mass markets.

The first principle refers to the observation that firms need not be inventors, but it is often useful to be first or early to market -- to be a “pioneer.” Many entrepreneurs and managers also need to think about more than one set of customers because markets can easily become saturated, and then firms will find their sales and growth rates collapsing. It is useful to target mass markets because these are where the money is. Moreover, directing the evolution of these mass markets to your company’s advantage is the best way to ensure that you end up with a good deal of the money!
Accordingly, Microsoft has either created or aggressively entered every major PC software mass market with products that are at least “good enough” initially to set de facto industry standards. These standards themselves then generate new markets that have helped Microsoft grow. (For example, the non-IBM version of MS-DOS created the market for IBM-compatible PCs. This has pushed sales of new computers as well as vastly increased demand for DOS and Windows applications products, which Microsoft and other companies provide. Microsoft’s on-line network may also vastly expand demand for on-line services, which Microsoft and many other companies will provide.) As indicated in the second principle, however, to preempt the competition, Microsoft development teams have continuously improved their products incrementally, focusing on particular features and producing periodic “upgrades” that customers continue to buy. Occasionally, Microsoft replaces old products with more advanced versions that make old products obsolete. (It is much better to make your own products obsolete than allow a competitor to do it.)

The third principle indicates that, to support product development, Microsoft has used another set of practices to help maximize volume sales. These are necessary to establish and maintain products as de facto industry standards. (This is a good way to ensure long-term profits.) The fourth and fifth principles reflect other tactics that Microsoft has used to encourage sales of its new products as well as to influence the direction that markets and the software industry overall take. More recently, it has been integrating, linking, and even simplifying new products to move into new markets as well as to expand their potential sales to hundreds of millions of home consumers.

None of these five principles is unique to Microsoft, although few firms have combined all these principles simultaneously in a single industry over more than one product generation. Gates and Microsoft seem to be unique in their ability to implement this combination of principles consistently. This ability, moreover, is a key reason for Microsoft’s persistent market power.

For example, creating or entering a potential mass market not as the inventor but as one of several pioneers, and then incrementally improving a “good” product that becomes the market standard, describes the behavior of Japanese firms in the VCR industry. Sony, Japan Victor (JVC), Matsushita, and Toshiba followed Ampex’s invention of the video-recorder in the 1950s to introduce a series of machines for the home. These firms also created a new market for video tapes. Microsoft’s use of high-volume exclusive contracts for MS-DOS and Windows resembles the way Japan Victor and its parent company, Matsushita, lined up high-volume contracts with original equipment manufacturers. These made VHS into the world VCR standard. Intel deployed a similar tactic to garner support for its microprocessors. Microsoft has also capitalized on its position as the standards provider to “push” new products into the marketplace, very much like VHS manufacturers and Intel have done (Cusumano, Mylonadis, and Rosenbloom, 1992; and Rosenbloom and Cusumano, 1987).

Not all industries involve platform standards (like an operating system, a microprocessor, or a video-recorder machine) and complementary products that must be compatible (such as computer applications software, computer hardware components, or video tapes). Platform technologies and complementary products enable their provider to “lock” customers and suppliers into architectural standards, components, and particular products. Some companies have taken advantage of this position for long periods of time, though not always “forever.” Again, we can cite Japan Victor and Matsushita with the VHS video recorder, and Intel with the
microprocessor. These companies are now under pressure from competing standards, such as digital systems and reduced instruction-set (RISC) microprocessors. There are other well-known examples of temporary market dominance based on product standards and complementary products: Xerox had the original plain-paper copier and various copier supplies and services. IBM sold mainframe computers as well as various hardware peripherals, operating systems, languages, development tools, and applications software. RCA dominated color television with replacement components as well as licensing fees. Sony sells the 8-millimeter camcorder and 8-millimeter tape cartridges. Nintendo and Sega dominate their industry with home video-game machines and their video-game cartridges and licensing arrangements. Sony and Philips control the standard for the compact audio disk and its extension to computer CD-ROMs.

Becoming the standards provider is not necessarily easy, though there are two more elusive challenges. One is to sustain this position long enough to take advantage of it with more than one generation of products. Another is to extend this dominance, such as through product or marketing linkages, to new markets. Though two decades old, Microsoft is still relatively young. It has been the top PC software company in sales only since 1988. It has dominated critical tools and platform technologies -- languages and operating systems -- for more than one generation (both character-based and graphical computing). But it has only dominated some applications segments. Nonetheless, Microsoft is one of those rare firms that has sustained and extended its market power. Two of the principles cited here explain how.

First, Microsoft frequently makes incremental improvements and occasionally introduces major advances in its products. While often doing little more than packaging many incremental innovations, these major changes make older product versions obsolete. With a continual cycle of incremental and occasionally more radical innovations, competitors have little opportunity to challenge the market leader. Microsoft has accumulated enormous financial and technical resources that enable it to sustain this level of R&D. It has also followed somewhat of an unusual strategy. Dominant firms generally hesitate to introduce new products that steal sales from their existing product lines (Henderson and Clark, 1990).

For example, Microsoft sustained the dominance of MS-DOS with Windows 3.1. This is both a graphical user interface and a set of new operating system functions. Windows 3.1 has made sales of MS-DOS alone and DOS applications largely obsolete. Windows 95 and Windows NT, and new applications written for these systems, make Windows 3.1 and older Windows applications outdated. Visual Basic (a graphical version of BASIC that greatly simplifies the writing of screens and menus for Windows applications) has replaced the traditional version of this language in the marketplace because nearly all new applications development is for Windows and other graphical systems. The Office applications suite, for both technical and pricing reasons, has practically eliminated the sale and use of individual applications programs. The Microsoft Network may even make obsolete some of Microsoft’s multimedia products now sold on CD-ROMs.

Second, Microsoft is continually integrating, linking, repackaging, and sometimes simplifying its products. The main objectives are to enter new markets with products that combine multiple functions that were once separate, and to make products more accessible to broader sets of users. This pattern of competition and innovation has enabled Microsoft to extend its reach to the enormous mass markets of the computer novice and the home consumer. Here, the potential sales numbers are every person on earth.
Again, this is not a unique strategy. VCR producers have combined their machines with TV sets as well as made the VCR and the camcorder extremely easy to use. What Microsoft is doing also resembles Apple’s efforts beginning ten years ago. Apple made the Macintosh computer extremely easy to use, bundled this with different applications, and has continuously offered new software products and even an on-line service. Nintendo, Sega, Sony, and other consumer electronics firms are extending the video-game machine to perform other functions. Telephone, cable-TV, and wireless communications companies are all developing new uses for their networks and thus extending their influence to new markets. Microsoft is doing the same thing: It is leveraging existing technologies and products as well as impressive capabilities in new product development. It is entering new but related mass markets by creating linkages among products and taking advantage of an enormous customer network.

**Gates the Leader and Manager:** As Microsoft has implemented this strategy of moving from one mass market to another, Gates has had to study constantly to stay on top of a fast moving company and industry, and remain knowledgeable about Microsoft’s expanding arsenal of products as well as those of competitors. He has been vastly underrated as a leader who inspires his employees and as a manager who has learned how to delegate authority. No one person understands the millions of lines of computer code in Word, Excel, Office, Windows NT, Windows 95, Internet Explorer, Microsoft Exchange, and other Microsoft products. The newer products also present whole new sets of concepts and technologies to master.

Gates has had able help, although he and other Microsoft managers have resisted creating large staffs for anything. Gates depends heavily upon the informal Brain Trust. He has only one personal administrative assistant and one technical assistant, hired during the past few years. The technical assistant is generally a young program manager or software developer who holds the job for a year or two. The assistant reviews product ideas and specifications, takes notes in meetings, follows competitors and trade shows, and helps Gates interact with projects, rather than isolating him.

Microsoft executives have introduced fairly conventional management systems, but Gates remains closely involved. He presides over program reviews and planning sessions that take place in April and October of every year. These sessions set the schedule for rolling out new products and establishing budgets. The October reviews center on three-year product plans. Each division explains the products it is planning to deliver and any interdependencies with other products. There is also a final session in which all divisions present at once to give everyone a common understanding of the product plans. After completing the October review, Microsoft’s marketing staff (called product managers) create a sales forecast based on the divisions’ product plans. Detailed budget planning then begins, based on product sales forecasts. Managers look at the sales versus budget estimates to determine how these compare with the profit objectives for the company. Based on this analysis, Gates and other top executives determine the head-count they want for the fiscal year beginning in June. Gates not only takes an active role in all the significant review and planning sessions, but he gives advice directly to the key product units while they are planning and developing new products.

Microsoft has also made minor modifications to its planning process since the arrival of the Internet and the faster pace of change that this technology has created. In 1998, the company still used the same three-year planning horizons and twice-annual reviews. But Gates also instituted monthly meetings of his executive staff to make sure they were on top of changes.
going on in the industry and able to make adjustments as needed. Microsoft executives also scheduled in special events, such as the “Internet Day” of December 7, 1995, as a mechanism to force managers to come to decisions about their strategy in a short period of time. Microsoft scheduled this day only a couple of months in advance, at the insistence of current president Steve Ballmer, in order to announce the company’s Internet strategy to the world even though, when Microsoft committed to the schedule, Gates and other executives had not yet decided what they wanted to do. Rather than challenge the Internet, Microsoft ended up deciding to embrace and integrate the Internet technology into everything that the company does (Cusumano and Yoffie, 1998).

In general, Gates concentrates on defining strategic new products or new versions of products, and keeping a check on development schedules, mainly through product status reports and electronic mail from project team members and managers. He receives short status reports from projects every month (these used to be bi-weekly), and actually reads them. He attends quarterly program reviews for many projects. He writes occasional strategic memos, perhaps four or five times a year. (One he was writing while we visited outlined “some of the big technical challenges we face and who's going to own particular solutions and what solutions are the right ones from a strategic point of view.”) Once or twice a year, he goes on “Think Weeks.” During these times, he isolates himself and thinks about a particular problem, such as how to improve customer support or get groups to cooperate more, or what a product should look like five years in the future. Gates also tries to “get the biggest bang for the buck” from the time he spends: “The products that comprise 80% of our revenue I choose to understand very, very deeply.” This means that he spends most of his time following the key applications such as Office, and its Word and Excel components, as well as Windows. He also closely follows new high-growth areas such as multimedia and on-line information services and products.

Each software product has a corresponding project status report. Projects send these each month to Gates and other top executives, as well as the managers of all related projects. They are a key mechanism for communicating between projects and top management. Gates usually responds to the relevant managers or developers directly by electronic mail, and then uses the information he gathers from these reports, and responses to his questions, for the formal program reviews.

The status reports are brief and have a standard format. Gates can read most of them quickly and still spot potential project delays or changes he does not want, although some projects and issues get more attention than others: “I read every one of those things.... It's not that hard to read a hundred status reports.... If it's an obscure mail gateway product, and if it hasn't slipped a whole bunch, then I just hit delete. The thing is very succinct.... It's like two screen-fulls.... There's a line for each date, like milestone dates, spec-by dates, code complete dates.... And then there's a column for the original date, the last date, and what they're reporting this time.” Gates especially looks for schedule slips, cutting too many product features, or the need to change a specification.

Microsoft also holds program review meetings for each project every three months or so. These meetings last about two hours, and are usually attended by Gates and other senior executives. Also coming by are “friends of the court.” Projects send one or two key people from each functional area: program management (the group responsible for writing down product specifications), software development (the people who write the computer code), software testing (the people who test the code), product management (the people in charge of product planning and marketing), and user education (the group that prepares product documentation). Gates focuses on the same types of issues he looks for in the status reports: Are projects “under control” and
anticipating problems? Do they seem to understand the design? Do they need to change the specs? Are they sharing code and managing interdependencies?

"You want to know is the project under control. That's fundamental. And you want to know if people are anticipating...major problems. If they're adding something that could potentially slow the product down, you say to them, 'Prove to me it's not going to be ten times slower.'... If something is taking longer to get done than they expected, was it because they just didn't understand the design? ... Maybe they should step back and redesign the approach they're taking. You have to ask enough questions to really understand are we on top of what we're doing. And you have to listen for ideas that have come up during the project that might be worth changing the spec to accommodate... [C]an [they] take something and push it to a higher level of generality? What's their relationship with all the other groups? .... I believe in code sharing across the groups. I've imposed on them the necessity to get this thing from this group and that thing from that group, and those groups don't work for them. If they want to point out to me some dependency that is creating massive bugs or it's massively slow or they may never get that piece from that group in time, it's important to bring that out in the discussion.... If the marketplace environment has changed and we're asking them to change the spec, it's important they hear from me about that. There's [also] a need to set a tone in terms of telling them they're doing super well or not doing super well, once you have the data. Sometimes you have that before you go into a meeting, because the amount of e-mail that's generated about projects is rather massive."

Gates tries to mix up his questions. He spends some time on obvious things and also probes more deeply: "I'll always surprise people with maybe half my questions, but about half the questions are fairly clear. I can remind people of which benchmarks really count. I can remind people of which system configurations really count." In some cases, he sends in some senior members of the Microsoft Brain Trust to help out:

"At my level, what do I really control? I have some very senior developers that I could shift over onto a project and have them help review the algorithms, review the code or just pitch in. So if a project really appears to be broken, then you want an independent review of the code. Very early in the company I'd say, 'Hey, give me the source code. I'll take it home.' I can't do that now. So I take somebody, a D14 or a D15 [these refer to Software Design Engineer Level 14 or 15, the top technical ranks in the company] and say, 'Go dig into this thing and let me know,' or, 'Help them out in terms of getting more personnel assigned.' They're always going to have a recommendation in terms of what features ought to be cut...because they understand all the dependencies and how the pieces fit together."

In reviewing specifications, Gates concentrates on a few key themes. He wants to know how “exciting” a new product is as well as how it fits with other Microsoft products. Increasingly, he asks about quality, mainly defined as bug levels. These three issues -- are there enough exciting features, does the product fit into the portfolio, and is the product stable in terms of bugs -- also drive Gate's recommendation on whether a product is good enough to ship. Another rising concern is managing “interdependencies” and the “integration” of products and product groups: making sure that groups cooperate and share common components, and that products slated to work together or share code are coordinated and on schedule.

Gates prefers to set broad directions and then leave the product groups to make decisions and solve these types of problems on their own. But he continues to astonish people with his ability to grasp the technical details of what the groups are doing. Preparing for an encounter with Gates can be intimidating even to seasoned employees and executives. Chris Peters, former Vice
President of the Office Product Unit, gave this advice to his colleagues in his now-famous 1990 video on “Shipping Software”: “You should keep Bill very informed about what you're doing and why. You should never hide anything from Bill because he's so good at knowing everything. But you should be firm and you should yell back. The only recommendation I can bring or give you guys is to bring your very, very, very best developers with you to the meeting so they can quote things off the top of their head and they can just bury him with facts.... Don't ever be unprepared.” (Peters, 1990).

To the question of what are the first things he let go of versus those he has insisted on controlling, Gates responded that he first let people write their own code without his rewriting it for them. Then he relaxed to checking specifications, key algorithms (mathematical or procedural recipes to accomplish particular tasks through computer code), and progress on schedules. He has retained control over five key aspects of new product development: (1) final decisions on important features in critical products; (2) approvals of new products that expand Microsoft’s portfolio; (3) coordination of groups building products that need to work together or share components; (4) monitoring of project progress, including decisions such as to add or subtract resources for the projects and product units; and (5) inputs into decisions on whether new products are ready to ship. Gates explained how his job has evolved:

“Well, first, I wouldn't let anybody write any code. I went in and took every statement that anybody else had written in BASIC and rewrote it myself, just because I didn't like the way they coded. That product had a certain craftsmanship thing. I was very reluctant to let anybody get involved in it. But then we had products like FORTRAN and COBOL, where all I did was make sure that we were designing to the right spec and review the basic algorithms.... The ultimate thing is what products do we develop. I have not delegated the general idea of products to develop. I don't come in and say ‘Well, I didn't know there's a whole new product group here. Nobody told me about that.’ That is a good decision for a CEO of a software company to keep in his hands. That's about the only one that I really control nowadays.”

A key role that Gates plays is to view the entire product portfolio of the company in light of the future directions he sees, including likely competitor moves. Then he makes the hard decisions: the technology versus business tradeoffs. Although Microsoft rarely “invents” totally new product concepts, Gates is usually quick to anticipate where markets and competitors are going. He also prepares his people to move fast once an opportunity materializes. Ed Fries, formerly the Development Manager for Word, recalled a visit to his group after Gates had just spent a week thinking about features and looking “five years” into the future:

“We interact with Bill a lot, especially in the early stages of a product when we're going over the spec and we're trying to decide what features we're going to put into the product.... Bill was here a few weeks ago and we walked him around and showed him all the stuff that we're doing in Word. He was on a 'Think Week' last week, and during that week he probably sent 10 pieces of mail to ChrisP [Chris Peters] that said, 'Oh, here's something I don't like about this version of Word. And here's something you should be thinking about in the future to make spelling better for all our products. Or Word and Help, and how can those things be combined into one type of thing.' So Bill does a lot to define our future direction with the product, and he also does some quick criticism of where we are now.... He always represents where he thinks things are going to be five years from now. We're trying to balance going in that direction with meeting the needs the users have today that we think we're not meeting. So we try to come up with a compromise of making that happen.... That's always the conflict.”
Building an Organization of “Smart” People

Smart people are the essence of the software business because computer programming is largely a creative intellectual activity. Bill Gates, as a former programmer of some note, clearly understands this, and decided to build a company consisting of technically excellent programmers and other managers with a good understanding of the technology. But he also wanted people who shared his understanding that software development should be a business -- not an “art” or a “hobby” -- for people who rely on this for their livelihood.

Thus, Gates decided only to hire people who were similar or complementary to him -- technically adept in an area useful to the software business, able to learn new things quickly, and able to see where market opportunities lay. His recruiting process has been rigorous and intensively personal, and Microsoft has continued this tradition of rigorous personal interviewing through to the present. As a result, it hires only a small percentage (around 3 percent) of people who apply to work for the company, and screens carefully for technical and business skills, even when hiring programmers. (This is in contrast to many companies building software systems in the United States, Europe, and Japan, which do not screen software developers so rigorously. It is typical in Japan, for example, for companies to hire people based on what university they attended, rather than on their aptitude for and experience with computers, and then train them in-house to write software. It is difficult for these people to perform well and be “smart” in a complex technology when they have such limited experience and training.)

“Smart” People: Gates and other managers frequently boast that they only hire “smart” people -- managers, as well as developers, testers, and others. But what does being “smart” mean? To Bill Gates, this means being able to understand and probe complex things quickly and creatively, as he explained in a 1994 interview: “There’s a certain sharpness, an ability to absorb new facts. To walk into a situation, have something explained to you and immediately say, ‘Well, what about this?’ To ask an insightful question. To absorb it in real time. A capability to remember. To relate to domains that may not seem connected at first. A certain creativity that allows people to be effective”(Playboy, 1994, p. 60). Gates himself exhibits these qualities and looks for them in potential Microsoft managers and employees. Gates also seems to include in his definition of “smart,” at least for developers, a deep understanding of software both as a technology and as a business opportunity.

As Microsoft has grown, Gates has personally interviewed hundreds of programmers, managers, and technical experts who complement and challenge his own skills and knowledge. From this group, he has cultivated a relatively small corps of senior people that serves as an informal Brain Trust to assist him in making decisions and overseeing critical projects or initiatives. Technical and business skills are useful to make complex decisions quickly and correctly, although there is a drawback. Gates has a tendency to promote programming experts into senior management at very young ages. Combined with very rapid growth, this has left Microsoft short of managers in the middle ranks who understand not merely the technology and the business but also the human side of management.
Microsoft's Brain Trust consists of around a dozen people. They run the key product areas and new initiatives as well as constitute an informal oversight group to critique what everybody else is doing. There is also a larger set of senior technical people working in the projects who are part of the broader Brain Trust network. Many of these people are Microsoft veterans from the founding years, although a growing number have come from competitors or are experts in new technical domains that go far beyond the PC.

**Technically Competent Managers:** In addition to the Brain Trust, Gates and other early executives in the company made a special effort to populate Microsoft with technically competent managers, especially but not exclusively in the development organization. Rick Rashid, Microsoft's VP for Research and a former professor of computer science at Carnegie-Mellon University, recalled being struck by the senior people he met: “One of the things that I was very impressed with when I first visited Microsoft was the level of expertise and competence that you find basically all the way through the management hierarchy. It's just not that common in the computer industry to find strongly technical people in key management positions throughout the company.”

Dave Thompson, who had BA and MA degrees from Cornell University and worked in DEC and Concord Communications for 13 years before joining Microsoft in 1990 to head the Window NT network design group, echoed this sentiment: “I believe that one of the strengths of Microsoft is the fact that its managers are very technical, even at the most senior levels.... I believe that managers should be as technical as possible. Leads, that is, first-level line managers, absolutely should write code, should be able to fix problems in the stuff that their people work on.”

We can see this technical competence reflected in how managers work. It is common practice for managers in all the functions at Microsoft to continue doing their functional jobs after becoming senior managers. Even development managers of large groups such as Word and Windows NT still spend a third to half of their time writing code. The rationale is that managers need to remain directly involved in the technology in order to make the right decisions and solve problems that occur in their groups. For example, Lou Perazolli, Software Engineering Manager on the Windows NT 3.0 project, spends about half his time coding. He makes managers under him do the same: “I have these rules in my group that I try to follow. Number one is everybody who works for me actually has code responsibilities. So nobody just manages people.... I find that people who manage people lose sight of what the goal is and they don't appreciate the problems and issues...and they don't act on them quickly enough. Yet when you have code that you are doing you tend to act on issues in a much more timely fashion and you tend to be more concerned about how things are going.... I probably spend 50% of my time looking at problems and writing code.”

Managers in the Applications area, such as Vice-President Chris Peters (speaking when he was the General Manager of the Word unit), have similar values:

“At Microsoft...the functional managers still spend a great deal of their time doing whatever they do. In other words, the development manager of a group of 60 developers will still spend a fair amount of his time programming.... In larger organizations...they keep pushing down the level where people are actually doing the work. Pretty soon you get a development manager who decides that he's way too important to program, and pretty soon the feature team leads all decide that they're way too important to program. Pretty soon the programmers are way too busy to program.... I have 270-some people working for me, and I'm the first layer in the organization that actually doesn't have to do any work, although I did write one feature in the new version.”
**Pluses and Minuses of “Smart” People:** Below the product groups and functional managers, Microsoft has approximately 4,000 software developers and test engineers. In a field where the best software programmers may write 10 to 20 times more code in the same time as the least productive members of the same team, attracting talented people is a great advantage. It means that the apparent resources available to managers with a high percentage of talented people are far greater than simply the number of bodies they command.

In fact, nearly every Microsoft manager interviewed for this study emphasized the value of having a highly selective pool of people. For example, former Executive Vice-President Mike Maples, who joined Microsoft after working for IBM, noted that, “The quality of the people is an unmeasurable benefit. Not everybody has the luxury we have of getting Mitt’s best and Stanford’s best.”

Rick Rashid agreed: “There are more good smart software developers here than I’ve seen anywhere else in the world, concentrated in one place.”

Dave Thompson agreed: “The...big difference between us and other companies is the people that we hire. Our whole system is predicated on the fact that our people are very fast and smart.... It makes you feel like you have a much larger set of resources to work with. You feel like you have flexibility when you have people that can fix things in hours instead of days.”

Bill Gates boasted that his developers are all “top-notch,” they all get involved in designing as well as actual coding, and they perpetuate themselves:

“We benefit from having the smartest people actually get involved not just in the design and ideas but actually the code itself. They know the code extremely well, and large areas of the code. It's good to have one person who can think through any kind of design change, particularly when you're late in the process and you're trying to be very careful. They can review anything that goes on. They've got a very strong model in their head for what... side effects that change might have.... We have the benefit that people don't like to have bad developers around. It's just enough of a drag on them that they'll find a way to convince people it's a bad match to be in here if you're not a really top-notch developer.”

In addition to providing an abundance of sheer technical skills, smart people can make even a large company seem relatively non-bureaucratic and flexible -- more like a small company. Chris Peters held this view, emphasizing the value of people who can work and solve problems on their own: “This is not a company where there's a bunch of stupid people running around and then some manager writes down a bunch of stuff in order to control stupid people. It's a company where there's a lot of smart people running around all trying to do the right thing.... I've seen stupid companies where they just hire bodies and attempt to make up for their hiring of lots of bodies by putting in lots of rules. I guess it may partly fix the problem, but the root cause of the problem was not lack of rules. It was hiring people that needed lots of rules to do their job.”

Given the type of people that Microsoft screens for, it is not surprising that the company culture emphasizes technical competence and shipping products -- rather than adhering to rules and regulations, or respecting formal titles, or cultivating skills in political in-fighting. Dave Moore, former Microsoft Director of Development, commented on this: “[T]itles aren't important in the company. What's always been important is shipping products -- how many products have you shipped?”

Senior Vice President Brad Silverberg added that authority and responsibility tend to move toward those people who demonstrate the most competence: “[A]nother key point about Microsoft development is that the balance of power within each group is more a reflection of the individuals and the strengths and capabilities of those individuals than any cookie-cutter
We're flexible. If I hired a development manager who was a super, super great product guy, who knew how to make features, balance of power can go over there. I'm certainly not religious about it. I just adapt to the strengths of the people I have."

If a conflict emerges between individuals or groups, such as over which components to share, Gates often steps in as the final arbiter -- the "smartest" of the smart guys. John Fine, formerly a Director of Program Management and Group Program Manager for Excel, offered this view of Microsoft's culture and the role of Bill Gates:

"Things get done around here in many ways, from the most informal ear whisperings to the most formal orders.... Rules don't count.... Politics does not reign supreme here. So it is certainly not unthinkable for, say, the business unit manager of one piece of software to talk to the business unit manager of another piece and say, 'You know, maybe our software should be inside of your software,' or something like that. That is not a political gaffe. It's just another tactic in order to get the job done of benefiting the industry the most, therefore making the most money. But because those decisions are very costly in opportunity and have such a huge potential for negative effects if they're wrong, of course, they'll end up with Bill. In that way, the traditional management tree is used." 20

The negative side of hiring smart people who do not want or need rules to work is that these people often have to learn by painful trial and error. For example, Microsoft developers and testers (with some exceptions) did not seem very well read in the software-engineering literature and often "reinvented the wheel," so to speak. Other companies had discovered good practices for software development years ago that Microsoft people are only now coming to recognize as important. (These include doing more careful design reviews and code inspections as well as paying more attention to architecture and designs, sharing components, and having better quantitative metrics and historical data for project management.)

It is also difficult for senior managers (including Gates) to tell "smart" people what to do. They do not always like to cooperate and compromise, or share what they create and learn. Microsoft has made some progress on this dimension of sharing and learning as an organization. But there is much more that the company can do. Mike Maples gave one example of the problems he saw when he first joined Microsoft:

"There weren't many people who had experience in project management. Things were just done loosely. The Microsoft people and the Microsoft culture was such that you had very smart people. They don't want to be told what to do, but they're willing to be allowed to discover what to do. And so you figure out the approach to let people discover what's a better way. 'You've got to have this process and this is the way you're going to do it,' and so forth, would never work. But facilitating people sitting down and talking about it and working it out [does].... That's a reasonable, good modern management style, if you've got capable people. You don't want too much of an authoritarian management.... I think we have a unique process that works better here than it would in most [companies].... But it is tailored to the fact that we have very, very smart people who are very, very motivated."21

As Microsoft has grown, it has introduced a number of policies to prevent groups from continually reinventing solutions to the same problems. But these remain guidelines or principles more than rigid rules, for the most part. Brad Silverberg commented on this transition within the company: "Back in the early days you didn't really need rules, but the practices have adapted to fit the new needs. But I think the culture is very strong, firmly entrenched. Decentralization, keeping
things as flexible as possible but not too flexible. There are some rigid points, but each group adapts to the rules a little differently.... Dave Moore or Mike Maples doesn't come to me and say, 'You are going to run your development organization this way.' It wouldn't work."

**Balancing Structure and Flexibility**

Good software programmers are usually creative, independent people who combine aspects of an artist with elements of an engineer or scientist. Like artists and scientists, they require considerable freedom to “invent” and “innovate” as they go along in a project. But they also require some subtle structure to make sure that they ship products to market in a reasonable time frame, rather than continuing to experiment and invest. Entrepreneurs and managers in software companies, therefore, must understand how to achieve this delicate balance -- how to organize creative people and help them structure their work. Then, if a company is successful, an entrepreneur must understand how to bring in more structure and systems to make sure the company can continue to develop, sell, and support products effectively.

Bill Gates not only understands the mentality of software programmers (because he is one), but he has also understood how to grow the company. He has learned through experience how to provide or allow just enough structure to make Microsoft work like a well-oiled, but still relatively flexible and creative “machine.” He has grown as a manager and learned what to delegate and what to control in order to remain an effective executive and develop other general managers to help him run the company. Microsoft has also matured considerably as a company and product groups have learned how to combine structure and flexibility in their development processes.

**Organizational and Process Evolution:** Bill Gates insists that Microsoft’s “dominant organizational theme is by products.” This is largely true, but Microsoft also pays a lot of attention to (albeit overlapping) technical or functional specialties. People in these jobs work in multi-functional teams organized by product, with some mechanisms to integrate across the product groups. Furthermore, in addition to divisions and product groups, Microsoft has two other organizational structures. One is formal, consisting of the management hierarchy. The second is informal. This consists of the loosely defined Brain Trust executives and network of technical people and managers who work on special assignments or projects, often at the suggestion of Bill Gates or other senior executives.

Microsoft adopted its organizational structure and process for product development in stages and through trial and error. During the early 1980s, it created an End User Group to develop applications programs. This allowed the systems and applications groups to evolve independently. The company struggled, however, to improve functional expertise, especially in testing. It also had to overcome problems stemming from a lack of focus in the product groups, and relatively poor mechanisms for quality control and project management. As in many other companies, several turning points and key actors defined this evolution (Table 2).
Table 2: KEY EVENTS IN MICROSOFT’S ORGANIZATIONAL EVOLUTION

1984  Establishment of separate testing groups in each division.

Recall of Multiplan 1.06 spreadsheet for the Macintosh.

Establishment of technical specialties (functions) such as Program Management and Product Management, in addition to Testing.

1986  Beginning of postmortem documents to identify quality and project management problems as well as potential solutions.

1987  Recall of the Macintosh Word 3.0 word processor.

1988  Arrival of Mike Maples from IBM and the establishment of separate business units for each product group within the Systems and Applications Divisions.

Use of milestones in the Publisher 1.0 project.

1989  The May “retreat” and November “Zero-defects code” memo highlighting the important of daily builds.

1990  Completion of the Excel 3.0 project (1989-1990) only 11 days late, using key elements of the synch-and-stabilize process (milestones and daily builds).

1992  Establishment of the collective Office of the President.

Centralization of Systems and Applications under the Worldwide Products Group, headed by Executive Vice President Mike Maples.

1993  Centralization of marketing teams (product managers) at the division level, except for product planners, and renaming of business units as product units.

Creation of the Office Product Unit, headed by Vice President Chris Peters.

First, was a decision in 1984 to set up testing groups separate from development. This provided an independent check on the work of developers. About this same time, program management began to emerge as a function distinct from product management and software development. Third, was a series of late and buggy products during the mid-1980s. This led to a growing consensus in the company that Microsoft had to become much more serious about quality control and project management. Microsoft groups now began documenting their experiences in projects through written postmortems, reflecting the belief that people could do a much better job at “learning from mistakes.” Fourth, was the arrival of Mike Maples in 1988 from IBM, and his decision to create smaller business units. These added more focus to the operating groups and made them easier to manage.

Fifth, was a 1989 retreat where key managers and developers grappled with how to reduce defects and proposed solutions that helped Microsoft teams become more systematic in software development and quality control. The discussions brought together critical ideas that had existed
before in the company and would become the essence of what my co-author of Microsoft Secrets and I call the *synch-and-stabilize* process for product development. One was the idea of breaking up a project into subprojects or milestones, which Publisher 1.0 did successfully in 1988. Another was to create daily “builds” or working prototypes of products, which several groups had done but without enforcing the goal of striving for zero defects. The project that received the most attention in Microsoft was Excel 3.0 in 1989-1990. This was the first project that was large in size and a major revenue generator for the company to use the new techniques, and it shipped only 11 days late.

Sixth, was the 1992 establishment of the four-person Office of the President and centralization of all product development responsibilities under Maples. These moves brought more structure to the operating systems groups, whose schedules remain difficult to predict. And seventh was a 1993 reorganization that centralized most product managers in marketing groups for the individual divisions, except for a few product planners who remained with the product development groups. At this time, Microsoft also renamed the business units as product units to reflect this change, and created the Office Product Unit. These changes have more clearly separated marketing from product planning and development, and facilitated sharing of common components across Microsoft’s key applications products.

Splitting Microsoft into the Systems and Applications Divisions came under former Microsoft President Jon Shirley, an “MIT dropout” and Tandy Corporation (Radio Shack) executive that Gates recruited in 1983. Gates had assumed developers should report to developers, not general managers, and that, as the top developer in the company, all development should report to him. Gates also tried to control other key areas, such as marketing. This approach worked when Microsoft had only a handful of products, but it quickly became obsolete. Shirley concluded that Gates had over-stretched himself and that a reorganization was necessary. In particular, Gates had a habit of pulling developers from one team and putting them on another, and making large changes in specifications with no prior warning or planning for such contingencies. To Shirley, the more appropriate role for the CEO was to define products at a high level of abstraction and set strategic directions for the development organization. He should not be bogged down in the details of each project (Ichbiah, pp. 182-183). Gates agreed to the reorganization and to limit himself to overseeing Applications, and he put Steve Ballmer in charge of Systems.

Bugs in different products prompted complaints from hardware manufacturers who bundled Microsoft’s operating systems with their PCs. (Microsoft refers to these companies as original equipment manufacturers or OEMs.) In response, Microsoft decided in 1984 to separate testing from development. For example, the version of BASIC that Microsoft shipped with the IBM PC in 1981 gave the wrong answer when the user divided “.1” and other numbers by 10. After this incident, Gates’ biggest customer of BASIC and DOS, IBM, insisted that Microsoft improve its processes for software development and quality control. Microsoft had other non-publicized problems in the early 1980s as well, such as a bug in its FORTRAN product (a technical programming language) that corrupted data (Manes and Andrews, 205, 231, 317, 366; Ferguson and Morris, 70).

At this time, individual customers also began complaining about quality problems in Microsoft’s applications products, which they could buy in retail stores. When Microsoft primarily sold languages and operating systems to OEMs such as IBM and Compaq, it relied on them to test its products. The OEMs would find bugs, and Microsoft would try to fix them in the next releases of its products. But when Microsoft began selling large amounts of software to individual
consumers, senior managers finally saw the need to introduce better testing and quality-control practices in-house. There was resistance internally, since most programmers and even some senior managers (such as Ballmer) insisted that developers could test their own products, assisted on occasion by high school students, secretaries, and some outside contractors. Microsoft did make a special effort with its new version of the Multiplan spreadsheet for the Macintosh by hiring Arthur Andersen to test the product before this shipped in January 1984. But an outside company usually cannot test a complex software product adequately. A serious data-destroying bug forced Microsoft to ship an update to Mac Multiplan's 20,000 buyers at the cost of $10 each -- $200,000 in total (Manes and Andrews, 248-249).

Microsoft managers concluded that they could not meet higher quality standards without setting up an independent testing group, following the lead of IBM and other companies with longer and more successful histories of software development. Dave Moore recalled this realization: "We knew that we couldn't stand alone with development doing their own testing. We needed a separate group that would build the test, run the test, give some feedback to development. That was a turning point." Microsoft did not copy IBM techniques completely, however. For example, it did not make large groups of people review all software items, from specifications documents to code and test plans, or require executives to "sign off" at the various development stages. These are "bureaucratic" practices common in software production for mainframe computers and defense applications but still rare in the PC world. Nor did Microsoft start monitoring in detail how developers and testers spend their time, as IBM and a few other companies do (such as Japanese software factories). Instead, Microsoft people selected what seemed to be good techniques, such as a separate testing group and automated tests, and code reviews for new people or critical components. Then they promoted these not as IBM practices ("That's not the way to get it used in Microsoft," noted Moore) but as "good practices" that worked.

But then, according to Moore, "We did it wrong." After Microsoft expanded the new testing groups during 1984-1986, developers "got lazy" -- they thought they could "throw code over the wall to testing." Yet they soon realized that developers find more of their own bugs than testers do, and that only developers can prevent errors from happening in the first place. Meanwhile, Microsoft shipped the next big disaster in company annals, Word 3.0 for the Macintosh, in February 1987 (after promising it for July 1986). Mac Word 3.0 had approximately 700 bugs, several of which destroyed data or "crashed" the program. Microsoft had to ship a free upgrade to customers within two months of the original release, costing more than $1 million (Manes and Andrews, 329).

By now, it had become apparent even to skeptics within the company that Microsoft was having enormous difficulties managing product development, and that groups would have to become more systematic to satisfy customers. Gates himself took over Applications, but several key projects remained in chaos. None of the new applications for Windows, except for Excel, were progressing. A database program (dubbed the Omega project, which evolved into Access) and a project-management application for Windows were in serious trouble.

Rather than let more projects spin out of control, Gates decided to go outside the company for more management expertise. In July 1988, he brought in Mike Maples, who had been with IBM for 23 years and headed software strategy and business evaluation. He was also a central figure in the development of OS/2 and an early graphical interface product for the IBM PC, Presentation Manager (Manes and Andrews, 370-371). It is somewhat ironic that Microsoft turned to a middle-aged manager from IBM for help. The IBM PC had been a success, although Presentation Manager and the OS/2 project with Microsoft had not gone well. Microsoft people also criticized IBM for
hiring too many people that were not very skilled developers (referred to disparagingly as "masses of asses programming"), and using a development process that was too sequential and compartmentalized (Manes and Andrews, 316). Microsoft managers also believed that IBM required thousands of people to do what Microsoft could achieve with just a few hundred top-notch developers.\(^\text{25}\) But Maples seemed to be a uniquely talented IBM manager, and Gates wanted to see more processes in place to make sure Microsoft built and delivered its products and controlled their quality more effectively. A key 1989 memo summarizing discussions from the May retreat galvanized the product groups to take corrective actions. The memo, titled "Zero-defects code" and written by Chris Mason, a development manager in the Word group, reflects the state of affairs and the new process Microsoft would adopt:

**Microsoft Memo**

To: Application developers and testers  
From: Chris Mason  
Date: 6/20/89  
Subject: Zero-defects code  
Cc: Mike Maples, Steve Ballmer, Applications Business Unit managers and department heads

"On May 12th and 13th, the applications development managers held a retreat with some of their project leads, Mike Maples, and other representatives of Applications and Languages. My discussion group investigated techniques for writing code with no defects. This memo describes the conclusions which we reached.... There are a lot of reasons why our products seem to get buggier and buggier. It's a fact that they're getting more complex, but we haven't changed our methods to respond to that complexity.... The point of enumerating our problems is to realize that our current methods, not our people, cause their own failure.... Our scheduling methods and Microsoft's culture encourage doing the minimum work necessary on a feature. When it works well enough to demonstrate, we consider it done, everyone else considers it done, and the feature is checked off the schedule. The inevitable bugs months later are seen as unrelated.... The reason that complexity breeds bugs is that we don't understand how the pieces will work together. This is true for new products as well as for changes to existing products.... I mean this literally: your goal should be to have a working, nearly-shippable product every day.... Since human beings themselves are not fully debugged yet, there will be bugs in your code no matter what you do. When this happens, you must evaluate the problem and resolve it immediately.... Coding is the major way we spend our time. Writing bugs means we're failing in our major activity. Hundreds of thousands of individuals and companies rely on our products; bugs can cause a lot of lost time and money. We could conceivably put a company out of business with a bug in a spreadsheet, database, or word processor. We have to start taking this more seriously" (italics added) (Mason, 1989).

These continuing problems with late and buggy products, in both the Applications and Systems Divisions, created a receptive environment for Maples to suggest changes. In particular, he wanted smaller groups that worked in projects targeting specific leading competitors and products, such as WordPerfect, Lotus 1-2-3, and Harvard Graphics. He also wanted each of these groups to define and then refine their processes for software development, testing, and project management. After discussing the matter with Gates, Maples broke down the Applications Division into five business units: Office, Analysis (Excel), Graphics (PowerPoint), Data Access, and Entry (Works). These now became independent profit centers, with self-contained resources for program management, development, testing, marketing (product management and product planning), and
user education -- very much patterned after the highly successful independent business unit that IBM used to launch the original IBM PC in 1981.

Microsoft has maintained this same basic organizational structure, with some refinements. In 1992, Gates moved Steve Ballmer from head of the Systems Division to head of the Sales and Support Group. He also created an expanded Worldwide Products Group, with Mike Maples overseeing product development for both Applications and Systems. This change gave Maples a chance to rein in the operating systems groups. Then, in 1993, Microsoft centralized marketing and created the Office Product Unit. In addition, Microsoft has appointed directors for the key functional areas: Development, Testing, Program Management, and User Education. Since 1994, this staff group reports to a new Director of Product Development Resources. The directors are either full-time or part-time. They do not oversee projects or have well-defined responsibilities, although they help groups identify and share “best practices.”

A clear benefit of Microsoft’s organization is that it provides the freedom for groups to operate as relatively small development centers totally dedicated to “shipping products” to particular markets. Chris Peters explained: “The business units, although we pretend that they’re mini-companies, are really pure product development centers.... There is no sales or finance -- all that stuff is removed from it. Everybody in a business unit has exactly the same..job description and that is to ship products. Your job is not to write code, your job is not to test, your job is not to write specs. Your job is to ship products.... When you wake up in the morning and you come to work, you say what is the focus? Are we trying to ship? Are we trying to write code? The answer is we are trying to ship.... You're trying not to write code. If we could make all this money by not writing code, we'd do it” (Peters).

**Product Development Philosophy:** How to coordinate the work of a large team building many interdependent components that are continually changing requires a constant and high level of communication and coordination. It is difficult to ensure that this communication and coordination take place while still allowing designers, engineers, and marketing people the freedom to be creative. Achieving this balance is perhaps the central dilemma that managers of product development face -- in Microsoft as well as in companies from many other industries. Dave Maritz, a former tank commander in the Israeli army who headed the MS-DOS/Windows testing group, commented on how he and other Microsoft managers try to impose only enough direction and iron-clad rules so that individuals and teams can work together toward the common goal of getting a new product out the door:

> “In the military, when I was in tank warfare and I was actually fighting in tanks, there was nothing more soothing than people constantly hearing their commander’s voice come across the airwaves. Somebody’s in charge, even though all shit is breaking loose.... When you don’t hear [the commander’s voice] for more than fifteen minutes to half an hour, what’s happened? Has he been shot? Has he gone out of control? Does he know what's going on? You worry. And this is what Microsoft is. These little offices, hidden away with the doors closed. And unless you have this constant voice of authority going across the e-mail the whole time, it doesn't work. *Everything that I do here I learned in the military.... You can't do anything that's complex unless you have structure....And what you have to do is make that structure as unseen as possible and build up this image for all these prima donnas to think that they can do what they like. Who cares if a guy walks around without shoes all day? Who cares if the guy has got his teddy bear in his office? I don't care. I just want to know ... [if] somebody hasn't checked in his code by five o'clock. Then that guy knows that I am going to get into his office.”

26
To accomplish this balance of subtle structure and creative freedom, since the 1989 retreat, most Microsoft groups have used two particular strategies. One strategy—focus creativity by evolving features and “fixing” resources—applies to product definition and organizing the development process. While having creative people in a high-technology company is important, it is often more important to direct their creativity. Managers can do this by getting development personnel to think about features that large amounts of people will pay money for, and by putting pressure on projects by limiting their resources, such as staffing and schedule. Otherwise, software developers run the risk of never shipping anything to market. This risk especially becomes a problem in fast-moving industries, when individuals or teams have unfocused or highly volatile user requirements, frequently change interdependent component designs during a project, and do not always synchronize their work as they go along.

Microsoft teams begin a project by creating a “vision statement” that defines the goals for a new product and prioritizes the user activities that need to be supported by the product features. Project managers (marketing specialists) take charge of this task, which they do while consulting program managers, who specialize in writing up functional specifications of the product. Next, the program managers, in consultation with developers, write a functional specification that outlines the product features in sufficient depth to organize schedules and staffing allocations. But the specification document does not try to decide all the details of each feature, or lock the project into the original set of features. During product development, the team members will revise the feature set and feature details as they learn more about what should be in the product. Experience at Microsoft suggests that the feature set in a specification document may change by 30 percent or more.

The project managers then divide the product and the project into parts (features and small feature teams), and divide the project schedule into three or four milestone junctures (sequential sub-projects) that represent completion points for major portions of the product. All the feature teams go through a complete cycle of development, feature integration, testing, and fixing problems in each milestone sub-project. Moreover, throughout the whole project, the feature teams synchronize their work by building the product, and by finding and fixing errors, on a daily and weekly basis. At the end of a milestone sub-project, the developers fix almost all errors that they, testers, and early users have detected in the evolving product. These error corrections stabilize the product, and enable the team to have a clear understanding of which portions of the product are complete. The development team may then proceed to the next milestone and, eventually, to the ship date.

Microsoft also structures projects into sequential sub-projects containing prioritized features, with buffer time (20 to 50% of total allotted time) within each sub-project to allow people time to respond to unexpected difficulties or delays, or add features during the project. Product designers create short vision statements (usually no more than a few pages) and outlines of specifications, rather than complete product specifications and detailed designs before coding, because Microsoft people have realized that they cannot determine in advance everything that they will need to do to build a good product. This approach leaves developers and program managers room to innovate or adapt to changed or unforeseen competitive opportunities and threats. Particularly for applications products, development teams also try to come up with features that map directly to activities that typical customers perform, and this requires continual observation and testing with users during development. In addition, most application product
designs have modular architectures that allow teams to incrementally add or combine features relatively easily.

Managers generally allow team members to set their own schedules but only after the developers have analyzed tasks in detail (half day to 3-day chunks, for example) and agreed to commit personally to the schedules they set. Managers then “fix” project resources by limiting the number of people they allocate to any one project. They also try to limit the time projects spend, especially in applications like Office or multimedia products, so that teams can delete features if they fall too far behind. (Cutting features to save schedule time is not always possible with operating systems projects, however, where reliability of the system is more important than features, and where many features are closely coupled and cannot be so easily deleted individually.)

The second strategy – do everything in parallel with frequent synchronizations – applies to the process of developing and shipping products. The objective here is to bring some discipline to the development process without trying to control every moment of every developer’s day. Managers in many different companies talk about making their companies less bureaucratic, more innovative, and faster to react through organization and process “reengineering” and “restructuring,” such as to speed up product development. But, as noted in this article, complex products often require large teams of hundreds of people, not small teams of a dozen or fewer engineers; and large teams can make communication and coordination extremely difficult and slow. Large-scale projects are simpler to schedule and manage if they proceed with clearly defined functional groups and sequential phases, and precise rules and controls. This approach, however, may excessively restrain innovation and underestimate the importance of synchronizing work frequently. Communication and coordination difficulties across the functions and phases may also result in a project taking more time and people to complete than projects that overlap tasks and make people share responsibilities and work in small, nimble teams.

What Microsoft tries to do is allow many small teams and individuals enough freedom to work in parallel yet still function as one large team, so they can build large-scale products relatively quickly and cheaply. The teams also adhere to a few rigid rules that enforce a high degree of coordination and communication.

For example, one of the few rules developers must follow is that, on whatever day they decide to check in their pieces of code, they must do so by a particular time, such as by 5:00 PM. This allows the team to put available components together, completely recompile the product source code, and create a new “build” of the evolving product by the end of the day or by the next morning, and then start testing and debugging immediately. Another rule is that, if developers check in code that “breaks” the build by preventing it from completing the recompilation, they must fix the defect immediately. (This actually resembles Toyota’s famous production system, where factory workers stop the manufacturing lines whenever they notice a defect in a car they are assembling. See Cusumano, 1985.)

Product teams also test features as they build them from multiple perspectives, including bringing in customers from “off the street” to try prototypes in a usability lab. In addition, nearly all Microsoft teams work on a single physical site with common development languages (primarily C, with some C++), common coding styles, and standardized development tools. A common site, programming language, and tools help teams communicate, debate design ideas, and resolve problems face-to-face. Project teams also use a small set of quantitative metrics to
track what is going on in a project and guide decision-making, such as when to move forward in a project or when to ship a product to market. In particular, managers rigorously track progress of the daily builds by monitoring how many bugs are newly opened, resolved (such as by eliminating duplicates or deferring fixes), fixed, and active.

**Lessons for High-Tech Entrepreneurs**

The case of Bill Gates and Microsoft should provide lessons for entrepreneurs and managers in many industries. What Microsoft does, however, is especially suited to high-tech companies operating in fast-paced markets with complex systems products, short life cycles, and competition based around evolving product features and technical standards.

For example, in terms of **strategy**, high-tech companies have to plan for rapidly changing markets and uncertain futures. Microsoft deals with this problem in two ways. On the one hand, Microsoft has a formal system of revolving 3-year plans that are reevaluated at least twice a year. On the other hand, Gates and his staff hire only excellent people and delegate to them a great deal of responsibility. Gates trusts his product unit managers to change plans and product designs as the environment or competitive needs change. High-tech companies also have to balance technical authority with strategic or business decisions. Microsoft deals with this problem by making sure that senior managers are either technically trained or very adept in understanding technical issues. And Bill Gates -- the most senior programmer in the company -- reserves the right to make final decisions on critical matters, and he remains closely involved with all key initiatives in the company.

In terms of **process management**, high-tech companies have to deal with issues such as balancing the need for rules and procedures to do complex work with flexibility and speed. Again, Microsoft achieves this by hiring very selectively and allowing individuals and teams a great deal of autonomy, and then imposing just a few rigid rules in product development -- such as the requirement that project teams create a build of the evolving product each day to synchronize changes that people have made in features or other aspects of the product.

Finally, in terms of **people management**, high-tech companies face the problem of how to recruit highly talented people and then cultivate both functional expertise (such as in programming, requirements writing, or testing) with business knowledge. Microsoft has dealt with this issue by devoting a large percentage of time of employee’s time to interviewing new recruits, hiring only the best people, assigning them clear functional roles in the company (program manager, product manager, developer, tester, etc.), but then requiring that they work in small feature teams and frequently synchronize their work with other teams. Managers also try to focus or “bound” the creativity of programmers by making them work on specific customer-oriented features with tight deadlines (at least internally) or limitations on the number of people assigned to projects.

It is another question, however, how long Microsoft will be around as a dominant company. It has a long way to go before it reaches the longevity of Ford, General Motors, or IBM, all founded in the early 20th century. Other global companies, such as AT&T, NEC, and Siemens, date back more than a hundred years. Nonetheless, Microsoft has already made a huge mark on history after merely two decades of existence. Competitors, government regulators, and not a few customers fear Microsoft will abuse its considerable influence. Probably, a combination of antitrust regulations and
market forces -- competition and user vigilance -- will keep Microsoft in check and allow the company to continue building better and cheaper products. As the usage of PCs becomes ubiquitous in the home as well as in the office, the world’s most powerful software company may yet become *the world’s single most powerful company* in any industry. This will be much to the chagrin of competitors. But it will clearly be to the credit of Bill Gates as an entrepreneur and manager, and to the credit of his senior management team and the Microsoft people.
REFERENCES


NOTES

1 This paper relies mainly on my co-authored book on Microsoft. See Cusumano and Selby, 1995. This study was
done between spring 1993 and summer 1995, and relied on approximately 40 in-depth interviews with Microsoft
employees as well as several thousand pages of confidential documents dating back to 1986. Microsoft cooperated
fully in the study, although it did not provide any funding. Microsoft also required the authors to sign a non-disclosure
agreement and removed some confidential data, although the authors retained full control over the final editing of the
manuscript.

2 Biographical details are based primarily on Manes and Andrews, 1993, with the quotes found on pp. 15-19. Other
sources include Ichbiah, 1993; Wallace and Ericson, 1992; and Cringely, 1993.

3 Interview with Steven Sinofsky, former Microsoft Technical Assistant to Bill Gates and current Group Program
Manager, Office, 8/3/93.

4 Interview with Bill Gates, Chairman and CEO, Microsoft Corporation, 8/3/93.

5 Gates interview.

6 Gates interview.

7 Gates interview.

8 Interview with Ed Fries, Development Manager, Word, Microsoft Corporation, 4/14/93.

9 Interview with Rick Rashid, Vice President, Research, Microsoft Corporation, 4/12/93.

10 Interview with David Thompson, Development Manager, Corporate and Network Systems (Windows NT),
Microsoft Corporation, 4/15/93.

11 Interview with Lou Perazolli, Software Engineering Manager, Portable Systems (Windows NT), Microsoft
Corporation, 4/15/93.

12 Interview with Chris Peters, Vice President, Office Product Unit, Microsoft Corporation, 4/12/93.

13 Interview with Mike Maples, former Executive Vice President, Microsoft Corporation, 4/16/93.

14 Rashid interview, 4/12/93.

15 Thompson interview.

16 Gates interview.

17 Peters interview, 4/12/93.

18 Interview with Dave Moore, Director of Development, Microsoft Corporation, 9/29/94.

19 Interview with Brad Silverberg, Senior Vice President, Personal Operating Systems, Microsoft Corporation,
8/4/93.

20 Interview with John Fine, Group Program Manager, Excel, Microsoft Corporation, 8/4 and 8/5/93.
21 Maples interview.

22 Silverberg interview.

23 Gates interview.

24 Moore interview, 4/14/93.

25 Silverberg interview.

26 Interview with Dave Maritz, former Test Manager, MS-DOS/Windows, Microsoft Corporation, 4/15/93.