PROJECT MANAGEMENT

CASE STUDY #1: Microsoft Office Business Unit (OBU)

Due: Oct. 3 @ 21:00h

READINGS:

Microsoft Corporation: Office Business Unit, Harvard Business Review

DISCUSSION/EXERCISE:

Objectives:

- Introduce students to real world problems
- To experience synthesizing a broad view subject into a one page executive style memo that clearly and effectively discusses critical points.

Assignment:

1. Individually, from a project management view, what were the top issues involved with the development of MS Word with regards to:
    a. History of MS's development process
    b. Stakeholders
    c. Market
    d. Technology
    e. Business drivers
    f. Culture(s)
    g. Time/schedule
2. The paper will be 1 page maximum, with a minimum of 12 pitch font, and 1 inch margins (use the word header/footer section for your name, assignment name, etc). Consider the paper from the view of senior management that has limited time to digest what you will write. You will probably not be able to include all of the above in 1 page, so pick the most critical items that impacted the Office Business Unit. If you write less than 1 page they you probably have not properly answered the question.
3. Spelling, grammar and "readability" count in the grade: **both Portuguese and English can be used**.
4. Be prepared to discuss the case in class immediately after the due date.

Harvard Business School

9-691-033
Rev. May 31, 1994

Microsoft Corporation: Office Business Unit

On March 30, 1990, Jeff Raikes, general manager of the Office Business Unit (OBU), glanced out the window one last time as the rain began to fall softly on the Douglas Firs at Microsoft's beautifully landscaped Seattle campus. The peacefulness of the setting contrasted sharply with the turmoil of the software development projects in the applications division.

One recent project, Opus, had been particularly difficult. Opus, Microsoft's code name for the Word for Windows[1] word processor development, had finally been shipped on November 30, 1989 after over five years of development. With the completion of the Opus project, two major issues arose: (1) what the follow-on project should be; and (2) how the development process's speed and effectiveness could be improved.

Although the schedule slipped significantly from the originally projected ship date (see **Exhibit 1)**, Word for Windows (known internally as WinWord) received significant critical acclaim. It was Microsoft's first word processor to be rated higher than archrival WordPerfect by the influential computer journal, *InfoWorld*. Sales exceeded Microsoft's projections.

Microsoft History

Microsoft Corporation had its roots in a company called Traf-O-Data that Bill Gates and Paul Allen founded in 1973 when Gates was only 16 years old. Gates had started programming when he was in junior high school and, by the time he was in ninth grade, local Seattle companies were employing him as a programmer after school hours. Gates and Allen formed Traf-O-Data in an unsuccessful effort to market software that generated summary traffic flow statistics derived from a rubber tube strung across a road. The software ran on an Intel 8008, the first 8-bit microprocessor[2] ever developed.

---

[1] Windows was an operating system written by Microsoft for the IBM PC which provided an easy-to-use graphic user interface, allowed convenient data sharing and switching between different application programs, and enabled programmers to write larger applications than with DOS.

[2] A microprocessor was a single chip that contained all the essential circuitry for a computer's central processing unit (CPU). The 8008 chip was a predecessor of the Intel 8088 that was used in the IBM PC.

Professor Marco Iansiti prepared this case as the basis for class discussion rather than to illustrate either effective or ineffective handling of an administrative situation.

In 1974 Gates was graduated from high school and began attending Harvard College. He did not stay there long, however. Early in his college career, he wrote a BASIC interpreter[3] for the first commercial microcomputer, the Altair, developed by a startup called MITS. The entire development required only four weeks even though Gates had to write the program without even seeing the Altair, working only from an emulator that Allen had developed. Soon after the program was completed, Gates and Allen reached an agreement with MITS to sell Gates's version of BASIC and Gates dropped out of Harvard to work full time with their company, which they renamed Microsoft. The Altair computer took off and Gates's implementation of BASIC became a standard. Over the next several years, Microsoft developed programs for a series of computers, including the Apple II and the Osborne portable computer.

Microsoft's big break came in 1980, when IBM chose it to develop the operating system for its personal computer. The result was MS-DOS (Microsoft Disk Operating System), a copy of which accompanied virtually every IBM and IBM-compatible personal computer (PC) sold.[4] IBM also worked with Microsoft to develop a BASIC and several other computer languages. DOS and the languages gave Microsoft a large, solid customer base on which to grow. And grow it did. From 1980 to 1989, Microsoft annual sales increased from less than $1 million to over $800 million (see **Exhibit 2)**. The company's employment grew from 45 people to over 4,000. In 1988, Microsoft's sales passed Lotus Development Corporation's (the maker of the 1-2-3 spreadsheet), making Microsoft the largest personal computer software company in the world.

Profits and stock price increased as well. In 1989, Microsoft's return on sales was over 15%—the highest of any major software company. Microsoft also had an aggressive employee stock ownership plan, and it was estimated that over 200 Microsoft employees had been made millionaires by increases in Microsoft's stock value. Gates, himself, had become the microcomputer industry's first billionaire.

By 1989, the approximate size of the personal computer software industry was $9 billion. While sales growth in the industry had been spectacular in the early 1980s, growth slowed somewhat in subsequent years, from a 30% yearly growth rate in 1987, to a 15% rate in 1989. Competition intensified, as the number of products on the market increased. Moreover the products themselves were becoming increasingly complex, incorporating more advanced features, and integrating with a greater variety of hardware and operating system environments.

During the 1980s, Gates almost singlehandedly determined Microsoft's technical direction (Allen, his cofounder left Microsoft in 1983 for medical reasons). After seeing results of research involving a graphical user interface (GUI) performed at Xerox's Palo Alto Research Center (PARC), Gates became convinced that this type of interface would become the standard in the industry.[5] For this reason, Microsoft started to develop a similar interface for the PC. The resulting program, Windows, added an additional layer of software between DOS and applications programs. Windows provided a set of utilities that allowed applications programs to work together in a user-friendly graphical environment. First released in 1985, Windows was slow to catch on. Microsoft continued to improve the program incrementally, and by March 1990, Windows had gained significant momentum with approximately 2.5 million of the 40 million PCs installed, having Windows.

Although the languages (compilers and interpreters) and operating systems for PCs formed the core of its business, Microsoft soon began to move into the applications software market. In the

---

[3] An interpreter allowed a user to program the computer using simple English-like commands.

[4] In this case, PC refers to any IBM or IBM-compatible personal computer. Microcomputer is the generic term for any single-user computer.

[5] In 1984, Apple introduced the Macintosh which employed GUI concepts and was widely praised as being fun and easy to use.

1980's, software could roughly be divided into three categories. Operating systems were the programs (like MS-DOS) that controlled the low-level operations of the computer (such as reading data from disks). Compilers and interpreters translated computer languages that were made up of English-like commands into machine language (series of 1's and 0's) that the computer could understand. Applications were the programs that end users typically ran (such as spreadsheets and word processors) to do specific tasks.

In 1983, Microsoft became the first software company to develop software for the Macintosh, and, by 1990, Microsoft was the major applications developer for it. Microsoft Excel was the dominant spreadsheet in the Macintosh market (with approximately a 90% market share) and Microsoft Word for the Macintosh was its dominant word processor (with 65% unit share and 80% revenue share).

Microsoft was not able to achieve the same market dominance in PC applications as it had in its other markets. Both its high-end word processing software (PC Word) and spreadsheet (PC Excel) were the second most popular products in their respective markets. Although these were successful products, they had yet to overcome their entrenched rivals (WordPerfect and Lotus 1-2-3). On the other hand, Microsoft was the only software company to compete successfully in all three segments, and was one of the few companies to have more than one market-leading application. By 1990, applications products accounted for over half of Microsoft's revenue (see **Exhibit 3** for a list of Microsoft's major products and their market shares in 1990).

Microsoft's products were not all a result of internal development. Gates actively sought out small companies with advanced technology. In fact, MS-DOS was an extension of SCP-DOS, a product developed by Seattle Computer Products that Microsoft purchased and adapted to the IBM PC. Although adapting and extending a product developed outside Microsoft was quite common, Gates usually preferred to perform the development of strategic products in-house.

Microsoft Organization

In 1990, Microsoft was divided into two groups: applications and systems (languages and operating systems). The Applications Division was headed by Mike Maples, who reported directly to Jon Shirley, the president and COO. Under Maples, there were six groups: Applications Strategy and five business units. Four departments comprised Applications Strategy, which provided central resources to all the business units. The resources ranged from programming tools and common subroutines to a User Interface Laboratory where the process by which test subjects learned and used software could be observed and recorded. Because these central resources were available on a voluntary basis, there was wide variability to the extent that the business units and even individual projects used them (see **Exhibit 4** for a partial organization chart).

All the business units were organized similarly, each specializing in a particular applications area. Jeff Raikes was the general manager of the Office Business Unit (OBU) which developed and marketed all of Microsoft's high-end word processors (PC Word, Mac Word, and Word for Windows). Under Raikes, the departments were organized by functional responsibility. Quality Assurance (also known as Testing) tested all the software for bugs and errors in the documentation. User Education wrote the documentation. Development, under Chris Mason, was responsible for developing the software. Product Marketing and Program Management also had separate departments. The other business units were responsible for different applications such as spreadsheets and databases.

The business unit organization had been instituted in August 1988 in order to help the Applications Division manage its growth. Prior to 1988, the entire division had been organized on a

functional basis—so that there was only one department for each in the division as opposed to one for each business unit in the division. Raikes commented on the change: "At Microsoft, we have a process of on-going organizational change—it helps to maintain a small company feel and team focus."

Evolution of the Development Process

In Microsoft's early days, there were few colleges or universities that offered degrees in computer science and computers were relatively scarce. Finding programmers was difficult and, like Gates, most of the early Microsoft developers had little or no formal training in computers. Many had been trained in other fields (particularly in math and science), but had fallen in love with programming.

The development process in the early days was more informal and there was less emphasis on schedule methodology and software architecture. Perhaps because of their lack of formal training, most of these developers did not follow the highly structured software development methodologies created by the Department of Defense and large corporate MIS departments. They often developed software without a formal specification or design. In 1990, stories about legendary developers abounded:

> There was one guy who could type at 80 words a minute. That's pretty impressive, but what's really impressive was that he actually wrote code at that speed. He'd write a 10,000 line application in two days, then if it didn't work, he'd throw it out and start again from scratch. He'd go through this process two or three times, until he ended up with a working program. Not only did it take him less time to do this than if he had sat down and tried to think everything out in advance, but the program that resulted was better too. Because he had implemented the same program several times before, he knew how to avoid all the pitfalls so his code became very clean.

One of Microsoft's early developers described his own views about software design:

> I design user interfaces to please an audience of one. I write it for me. If I'm happy I know some cool people will like it. Designing user interfaces by committee does not work very well: they need to be coherent. As for schedules, I'm not interested in schedules; did anyone care when *War and Peace* came out?

Raikes described some of the problems of relying too much on such star performers:

> There are a lot of problems with relying on individual superstars: (1) they are in very short supply; (2) someone has to maintain and update the software they've written (which often doesn't interest them) and often other people have difficulty understanding their code; (3) sometimes they don't understand what the market wants; and finally if you try to put several of them on the same project you get real problems with design decision—"too many cooks spoil the broth."

During the early to mid-1980s, Microsoft was often criticized for writing software which was technically excellent but difficult to understand and hard to use. In an attempt to make Microsoft more responsive to the market, in the early 1980s Gates began to hire marketing specialists, both from other software firms and also directly from MBA programs. Many of the new arrivals were not technical experts, but their mission was to reorient Microsoft to focus on the customer.

To help bring a more coherent perspective, a program management function began to evolve. Jabe Blumenthal, a marketing assistant who became the first program manager, described the evolution of the function in more detail:

> In early 1984, we began to work on a spread sheet for the Mac. I got involved and became a sort of service organization for the development group. I helped document the specifications, do the manual reviews, and decide what bug fixes were important and what could be postponed to a later release. While I didn't make the design decisions, I made sure that they got made. The process worked out really well, so they decided to call it something and institutionalize it.

Raikes described the purpose of the program management function:

> Program Management was introduced to formalize design, coordinate product creation functions (development, testing, and user education), and perform support functions (such as manual reviews, and competitive product evaluation).

In 1990, several people shared leadership for the development of a new product: the **project lead** and the **technical lead** from the development group, the **program manager** from Program Management, the **product manager** from marketing, the **on-line lead** and **print-based lead** from User Education, and the **localization lead** from the international division. These people were supposed to work as a group, with no one person having total authority. The project lead was responsible for overseeing the product development effort including handing out programming task assignments, scheduling, and coordinating the development effort. The technical lead had final say in all technical decisions, code reviews, and programming standards. The product manager handled all the marketing issues such as competitive analysis, positioning, packaging, and advertising. The program manager's job was to integrate and coordinate the efforts of everyone involved in the project. Program managers were also directly responsible for the concept and specification of the product. The on-line and print-based leads handled the user education functions and the localization lead oversaw the customization of the program for various international markets.

A team of about a dozen developers was usually dedicated to each major development effort, and was responsible for writing the code. Microsoft managers were quite proud of the small size of their teams. Other major competitors would use much larger teams often including more than one hundred programmers for major applications. Microsoft's cost per line of code developed was significantly lower than the industry average, which was about $125 in 1989.[6]

Despite the fact that he had a key role in the development of the system, Blumenthal felt that the process had many problems:

> The problem is that they did not copy a lot of what caused our effort on the Mac spreadsheet to be successful. We had a small group, I regarded myself as a service organization, and I was best friends with the developer. Now program management owns the product; they write the spec and throw it over the wall. The product managers don't know much about the realities of development, and the developers don't know the competition. The program managers have become a bottleneck—they make all the decisions, and the developers act as if they have no responsibility for the product.

---

[6]Rules of thumb for programmer productivity in the industry averaged over a project typically ranged between 5 to 10 lines of code per day. These measures were considered very rough guidelines.

Chris Mason, the development manager of the Office Business Unit (OBU), however, described a different interpretation of the development process:

> While Microsoft has become somewhat more formalized, the basic truth is that developers still have ultimate control of the process. If they want to check in a piece of code, there is no way we can stop them. There comes a time in every development project where the development manager says 'no new features shall be added,' but if the developer feels strongly, there is not much the manager can do. Microsoft has a deep underlying philosophy that people know what they are doing and will try to do the right thing.

Despite the changes in the development process, Bill Gates remained a constant factor, actively involved in every major development project. He attended design meetings periodically, reviewed design specifications and project schedules, and read many of the periodic status reports. Although many people at Microsoft had experienced his sometimes harsh critiques, there was universal respect for his technical expertise and ability to forecast where the computer industry was heading.

The Development of Word for Windows

Microsoft introduced its first high-end word processor for the PC, "PC Word", in late 1983. The product received only lukewarm reviews, and its sales were mediocre by Microsoft's standards. In September 1984 Gates, convinced that a key opportunity was being squandered, prompted the development of a new revolutionary word processor. The new product was going to run on the Windows operating system (at that time in development), and was going to exhibit some extremely innovative features to make Microsoft the leader in PC wordprocessing. Gates gave three people, John Hunt, Andrew Hermann, and Lee Arthurs, responsibility for the project which was code named Cashmere. Greg Slyngstad, who later became program manager for Cashmere, describes the start of the project:

> Gates put three real hotshots on the project. Hunt, who had single-handedly written the first version of PC Word, became the project manager. Arthurs, who had a Ph.D. in psychology, was responsible for user interface and documentation. Hermann had been at Wang and was supposed to know the word processor business inside and out.

In giving the Cashmere team its assignment, Gates directed that they "develop the best word processor ever," and that they complete the project as soon as possible—preferably in less than a year. Thus, the project was scheduled to be completed by October 1985. Not much progress was made, however, in the first year of the Cashmere project. Hunt brought on a couple of software developers to prototype software and he, Hermann, and Arthurs wrote several papers suggesting the features to be included. Their original concept was to integrate a uniform user interface and data structures for multiple purposes at the lowest level. In other words, they planned to structure the program and data so that it integrated seamlessly into other types of applications such as spreadsheets and databases. Not only would the new product be able to interface with other applications, but it would include some of the same types of features that these applications included so that the distinction between product categories would blur. Some of the specific features to be included were electronic mail, document protection (so that other people would not accidentally destroy someone's documents), facilities to build mail lists, and primitive spreadsheet capabilities.

By the beginning of 1986, the scheduled ship date was still approximately a year away, and Gates began to put pressure on Hunt to get some visible results. Eventually the pressure got to be too much for Hunt and he left the project in July 1986.

When replacing Hunt, Gates decided to use the program manager concept which was being developed at the time. Three Microsoft veterans were brought in: Doug Kurtz, the PC Word development lead, took over the same role for Cashmere; Lars Dormitzer, a well regarded developer, assumed the technical lead role; and Slyngstad became the program manager. A new marketing manager, Jeff Sanderson, was also brought in. As a new team member recalled:

> We all thought that the project was much farther along. Although Hunt had written a bunch of papers describing features that he wanted, there was no comprehensive specification of what would be in the product. We ended up throwing everything that had been done out, and started from the code used in Word for the Macintosh. We were a year behind schedule from the first day we started.

At this point, the project was renamed Opus. A new team of developers was formed. Due to a shortage of experienced developers, the new team was almost entirely staffed with new hires. Very few of the developers had had experience with other Microsoft projects.

During the second half of 1986 and the first half of 1987, a lot of effort was spent writing a new specification for the product. As time passed, however, pressure built up on the team to show visible results. The schedule continued to slip into 1988, and the pressure increased to a disturbing level. Sean McDermott, who was a software development engineer (SDE) on Opus at the time, recalled this period:

> We were under a lot of schedule pressure. Some managers seemed to regard the schedule as a contract between them and the developers. Furthermore, when development came up with a new schedule, management questioned every estimate.

Upper management kept up the pressure. During one meeting in early March 1988, one manager indicated that he felt that the Opus team was the worst in Applications Development. Chris Mason, the OBU development manager, recalled the effects of the constant schedule pressure:

> Opus got into a mode that I call "Infinite Defects." When you put a lot of schedule pressure on developers, they tend to do the minimum amount of work necessary on a feature. When it works well enough to demonstrate, they consider it done and the feature is checked off on the schedule. The inevitable bugs months later are seen as unrelated. Even worse, by the time the bugs are discovered, the developers can't remember their code so it takes them a lot longer to fix. Furthermore, the schedule is thrown off because that feature was supposed to be finished. These problems aren't unique to Microsoft—every company in the industry faces them.

In April 1988, Dormitzer had to take a medical leave of absence. McDermott, who had less than two years of experience, was made technical lead. Although McDermott was relatively young and inexperienced for such a position, the job required a very detailed knowledge of the program, and no one with more experience had the required knowledge. McDermott was also an outstanding technical contributor. Two months later, Kurtz, who had tired of the continual pressure, took a leave of absence from Microsoft. Dormitzer, who had recovered somewhat, came back to take the development lead.

Over the next several months, progress was made on Opus. All the required features were coded (though not debugged) and the team declared that the "Code Complete" milestone was reached in October 1988. Code Complete meant that all that remained to be done was to debug and optimize the code for performance. This phase was called "stabilization," and once the code had stabilized (all known bugs were fixed and performance was adequate), the product could be shipped. For scheduling, Microsoft used a rule of thumb that the stabilization phase typically lasted three months.

It soon became clear that Opus was not likely to follow the three month rule. Although developers were fixing bugs quickly, the testers seemed to find new bugs just as quickly (see **Exhibit 5**). During this period, Dormitzer did his best to lead the project, but his illness had not fully abated. Eventually, Mason responded by making McDermott acting development lead in addition to his technical lead responsibilities. At that time, McDermott had been at Microsoft for three years.

McDermott recalled the stabilization period:

> Not having a technical lead who could concentrate on technical issues definitely cost us. The size, speed, and memory usage of Opus could have been made better than it was if the technical lead had not spent the last 18 months as development lead or covering for sick and burned out development leads. The team at this stage had 15 developers, six programmer assistants and seven interns during this period. No one lead could possibly keep tabs on what each person was working on.

Despite the troubles, the Opus program began to stabilize. During the spring of 1989 the number of active bugs remained relatively constant, but, during the summer, an initiative emphasizing the quality of changes rather than the quantity of changes was instituted. For the first time, testing was invited to development code reviews, and ownership of the code was stressed. By late fall of 1989, the program had stabilized and Word for Windows version 1.0 was released on November 30, 1989 (see **Exhibit 6** for a project time line).

Word for Windows' Market Reception

Despite the long delay in the Word for Windows (WinWord) development, only one other company, Samna, had been able to release a full featured word processor for Windows earlier. While it was too early to gauge the customer reception accurately, early signs were very encouraging. One particularly encouraging sign was the reception in the industry press. Reviews in computer magazines and journals wielded great power over the marketplace perceptions, and the reviews had been universally favorable. WinWord was described as easy to use and incredibly bug-free for a first release. Many magazines rated WinWord higher than any other word processor for the PC. (See **Exhibit 7** for a description of competitive products). In response to WinWord's success, WordPerfect announced that it was developing a word processors to run under Windows. WordPerfect for Windows was scheduled for release in February 1991.

WinWord Postmortem

While the WinWord development project had been an extreme case, its problems were not unusual at Microsoft (or for any software development project at any company). In order to learn from the mistakes of previous development projects, Microsoft had instituted a policy of reviewing every project on its completion. The review entailed gathering many statistics on the project as well as holding a series of meetings with project participants in which their views on the project were discussed. While the exact type of statistics varied somewhat from project to project, typically they

included estimated versus actual schedule over the course of the project, bug counts over time, code size over time, and milestones scheduled versus actual completion.

These statistics and summaries of the discussions from participant meetings were collected in a document called a postmortem, which was then distributed to all managers in the business group as well as to senior managers. Most postmortems were about 25 pages in length, but the Opus document ran over 100. Extensive statistics had been collected on the project (including those in **Exhibits 1** and **5**).

Ideas for Improving Product Development

There were many views on the best way to prevent scheduling fiascos like Opus from recurring. While some of the ideas focused on improvements in the development process, others focused on Microsoft's approach to project management, or on its development strategy.

Process

While the majority of developers enjoyed Microsoft's informal approach, some felt that a more structured development process would substantially improve the company's development performance. This would include relying on more formal project phases and strict milestones, as well as the implementation of formal structured methodologies for software development. Among the developers, McDermott and Mason were advocates of a more structured approach.

McDermott, having lived through the Opus project, had strong views about the causes of its problems:

> A major problem was the lack of an early, clear direction and specification. While we need some flexibility to respond to new information and market changes, from the time development work begins, the major features should be down.

McDermott believed that the development process should have two major clear phases: concept development and implementation. In the first phase, a product concept would be carefully investigated, leading to a complete specification. Once the specification was down, everyone would buy in and implementation would follow without interruptions or delays.

Mason had a different opinion. He had written documents proposing a different approach to improving Microsoft's software development process. He called it "Zero Defects":

> Our goal should be to have a working shippable product every day. This means that when a programmer says a feature is done, it is totally complete: all error and boundary cases work, all interactions with the rest of the product work, and test code and documentation have been checked in.

In this mode of development, the project would begin with a brief period of time in which senior team members would lay out the basic architecture of the software application, breaking it down into several sets of desireable features. These feature sets would then be prioritized. The software coding process would focus on each feature set in a sequential manner. The code would be written in "object-oriented" modules, each of which could be tested independently.

After each module was completed, work would proceed to the next module. This would allow the product to evolve in a controlled fashion, gradually adding subsets of features under the

scrutiny of the project's management. This would imply substantial flexibility in the product's specification, since feature sets could be added at any time before (or after) the shipping date. Conversely, this process would also allow meeting virtually any shipping date, with a clear tradeoff between time-to-market and available features. Among the major costs of this approach, however, would be sacrifices in speed and program efficiency: the modularity in the code would make the application substantially slower and larger (in memory usage).

Project Management

Some of the program managers at Microsoft felt that changing the structure of the development process would not be enough to improve things substantially. They felt instead that the root cause of Microsoft's problems was in its approach to project management. The current approach lacked focus and control.

Some program managers felt that their role and position in the organization ought to be strengthened substantially. Their role should evolve to one of "software designer," in charge of conceptualizing and specifying the full range of features and characteristics of the product to be developed. The developers would then simply implement their design in code.

Several program managers emphasized that this approach was consistent with the opinion of several industry experts who felt that the character of the best software applications had changed in recent years. They felt that developing outstanding software now required a much greater attention to detail rather than sheer performance: attention to how the individual features fit together into a well designed, "coherent" product that was attractive, reliable, and fun to use.

Most developers were strongly opposed to turning program managers into designers, however. With a few exceptions, current Microsoft program managers had not previously been developers. The developers thus felt that program managers simply did not have the level of technical sophistication that was required to really understand the potential of a software development project.

Development Strategy

Other managers emphasized Microsoft's lack of a uniform strategy at the business level as the fundamental problem. They focused on the need for coherence across similar applications, to create economies of scale in development as well as a common "look and feel" in the products.

In 1990, Microsoft was actively writing applications to run on several different operating systems: the main ones were MSDOS, MSDOS with WINDOWS, and the Macintosh. Independent teams would be dedicated to the development of an application for each system. For example, there was a team for Word for the Macintosh, a team for word for DOS, and a team on Word for Windows. Keeping up efforts aimed at different operating systems was putting considerable strain on Microsoft resources. A set of suggestions thus focussed on code sharing within the Applications Division. Different programs required similar components, such as drop-down menus, macro languages, and graphics. Previously written software could often provide similar functionality to a large portion of the routines developed in a typical project. This would create development efficiencies as well as a more consistent user interface across different programs.

On the other hand, as Mason described, sharing code on Opus had cost more time than it saved:

> On WinWord trying to share code with MacWord was a tremendous source of delay. We took a whole bunch of MacWord code at the very beginning—which

> was fine. The problem came when the Opus made slight changes to optimize the code for WinWord—or just changed it because they felt that the way the MacWord people had done something that was "brain-dead." Then the MacWord people would announce that they had found a whole bunch of bugs in the code they had given Opus. To get those fixes, Opus would then be required to make the same changes so that it was compatible with WinWord. This happened about three or four times. Not only did it delay WinWord by maybe six months, but it probably delayed the release of MacWord version 4.0 by about eight months. To share code, you can have only one set of source code and one group of people modifying it. Sharing 80 percent of the code in a project is good, 20 percent is a nightmare.

Recognition of the code-sharing problem had led to the "core code" approach. In this approach, almost all of the functionality of an application was written in "generic" code which would be shared by the teams working on each individual operating system platform. The Excel business unit had used the approach and Pete Higgins, the general manager, felt that the long term reduction of the development effort would be worth the substantial hiatus in product improvements. No new Excel application had been introduced for a full two year period. Raikes felt that there were also other drawbacks to this approach:

> A risk with the core code standardization approach is that the software tends to get written for a lowest common denominator platform. For example, if you are writing for a PC and a Macintosh, you will tend to use only the features that are common to both. This means that it's difficult to write software that uses all the capabilities of either. Because of this, you may not come out with the best software on either machine and the way the market is today, you need the best software to compete.

Options for the Next Version of WinWord

After a lot of thought, Raikes had boiled down future WinWord project possibilities into two basic options.

The first option involved introducing a new WinWord version (version 2.0) in as rapid a time period as possible. He was pleased with the product's market reception, but he felt that it could use some performance improvements, a few important bug fixes, and the addition of a few features. This was a common strategy at Microsoft. In many cases, it was only when the second version of a new application shipped that its sales really took off - many key customers would wait for the inevitable product improvements before committing to new software. The goal would be to ship WinWord 2.0 in slightly more than one year, or about when WordPerfect had announced it would enter the Windows market. This would involve a highly focused effort, and Raikes doubted that it could be combined with the major improvements in development process and strategy suggested above.

The second option was to postpone the introduction of WinWord 2.0, but use the project to substantially improve the product development process in his business unit. He could experiment with freezing the product's specification, as suggested by McDermott or with Mason's modular approach. Additionally, he could focus the team on "core code" development. Word for DOS, MacWord and WinWord would be rewritten so that 80% of the code was common and only a small part was machine specific. Raikes expected that working on these improvements would add substantial delays to the shipment of WinWord 2.0, ranging from one to two years. Perhaps most important, it would add substantial uncertainties, since these approaches were relatively novel at Microsoft.

In each option, a team of approximately ten programmers and four programmer assistants would be assigned to the project. This was Microsoft's standard team size and had been chosen because Microsoft's management felt that it was the largest practical team. Raikes was uncertain about having enough resources to emphasize more than one project, since there was a limited number of outstanding, experienced developers available.

As Raikes considered his options, his primary goal was clear—he wanted Microsoft to surpass WordPerfect and develop the best-selling word processing software in the world. Achieving such a goal would be difficult, but Raikes felt that if he chose the correct development option and managed the project well, it was possible.

Exhibit 1 Actual vs. Projected Schedule

Report Date	Estimated Ship Date	Estimated Days to Ship	Actual Days to Ship
Sep-84	Sep-85	365	2,187
Jun-85	Jul-86	395	1,614
Jan-86	Nov-86	304	1,400
Jun-86	May-87	334	1,245
Jan-87	Dec-87	334	1,035
Jun-87	Feb-88	245	884
Jan-88	Jun-88	152	670
Jun-88	Oct-88	122	518
Aug-88	Jan-89	153	457
Oct-88	Feb-89	123	396
Jan-89	May-89	120	304
Jun-89	Sep-89	92	153
Jul-89	Oct-89	92	123
Aug-89	Nov-89	92	92
Nov-89	Nov-89	0	0

Exhibit 2 Financial Information ($ millions)

	Fiscal Year Ending		
	6/30/89	6/30/88	6/30/87
Net Sales	804	591	346
Cost of Goods	204	148	74
Gross Profit	599	443	272
Research and Development Expense	110	70	38
Selling, General and Administrative	247	186	107
Non-Operating Income	9	(4)	(6)
Interest Expense	0	0	0
Income Before Taxes	251	184	121
Provision for Income Taxes	80	60	49
Net Income	171	124	72

Source: *The Wall Street Journal*, May 21, 1990, p. A4, reprinted with permission.

Exhibit 3 Microsoft's Product Line and Market Shares (estimated for 1990)

Product Line	Sales ($ millions)	Market Share[a]	Market Rank
IBM PC Operating Systems	$300	98%	1
MS-DOS, PC-DOS	130	91	1
OS/2	56	5	3
Xenix	14	2	4
Windows (requires DOS)	100	15	2
PC Languages	95	70	1
C	50	65	1
Pascal[b]	10	35	2
FORTRAN	35	92	1
PC Applications	430	9	2
Excel (windows spreadsheet)	120	15	2
Multiplan (spreadsheet)[c]	10	1	8
Word (word processor)	250	30	2
Word for Windows	50	6	6
Macintosh Languages	17	80	1
C	9	85	1
Pascal	2	55	1
BASIC	6	95	1
Macintosh Applications	140	25	1
Excel (spreadsheet)	70	90	1
Mac Word (word processor)	70	80	1

[a]Market share was defined as the percent of sales in the particular product category. For example the market share of Word for Windows was defined with respect to all PC wordprocessors (but not including Macintosh word processors). It should be mentioned that the market shares for PC operating systems total over 100% because all Windows systems must have DOS as well.

[b]Borland was the leader in this category with Turbo Pascal with approximately $20 million in annual sales and a 60% market share.

[c]Lotus dominated this category with 1-2-3 Release 2.2 with $230 million in sales and 1-2-3 Release 3.0 with $120 million in sales annually. Borland's Quattro Pro was also a player in this category with $60 million in sales.

Exhibit 4 Microsoft Organization Chart, 1990

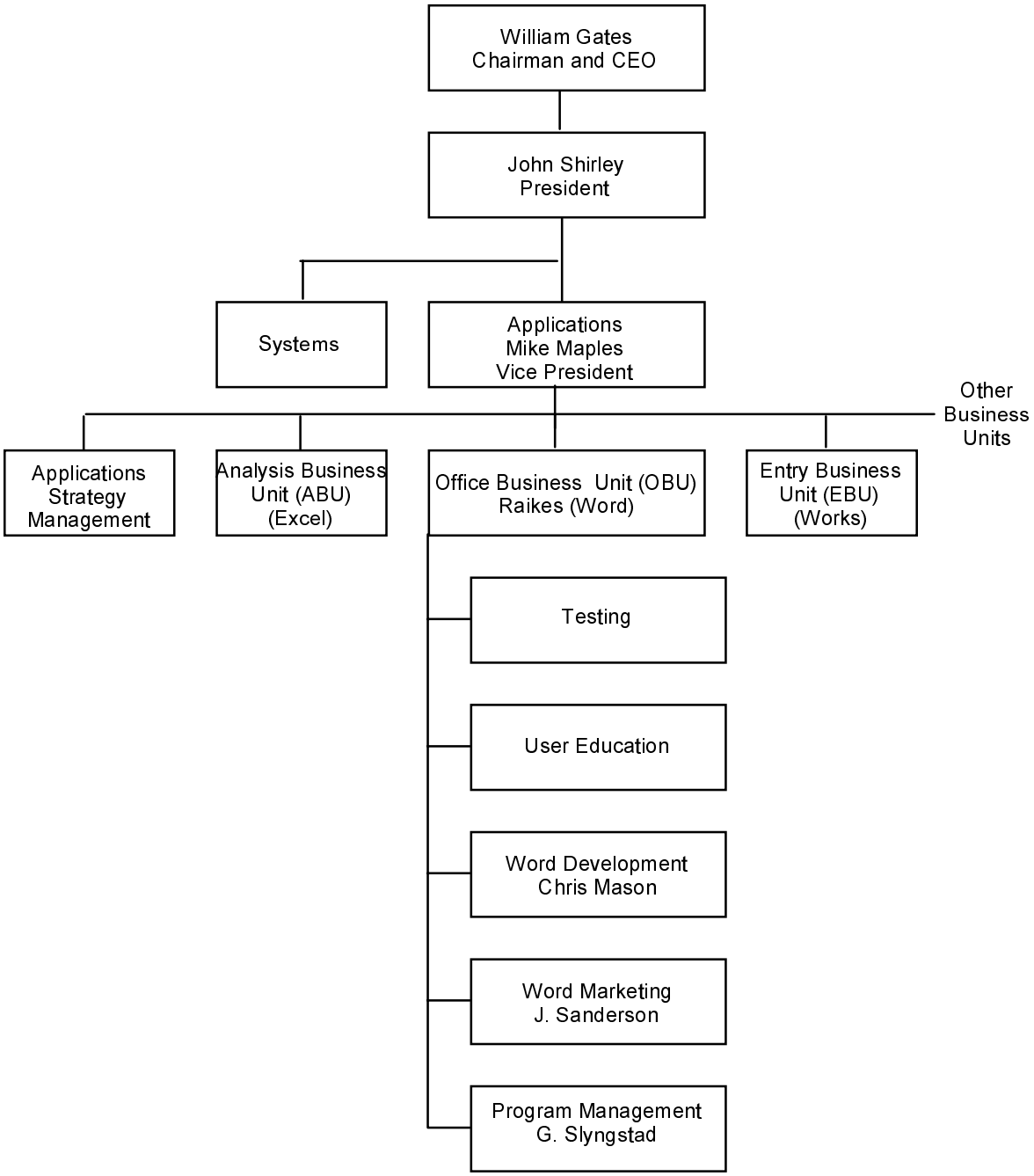


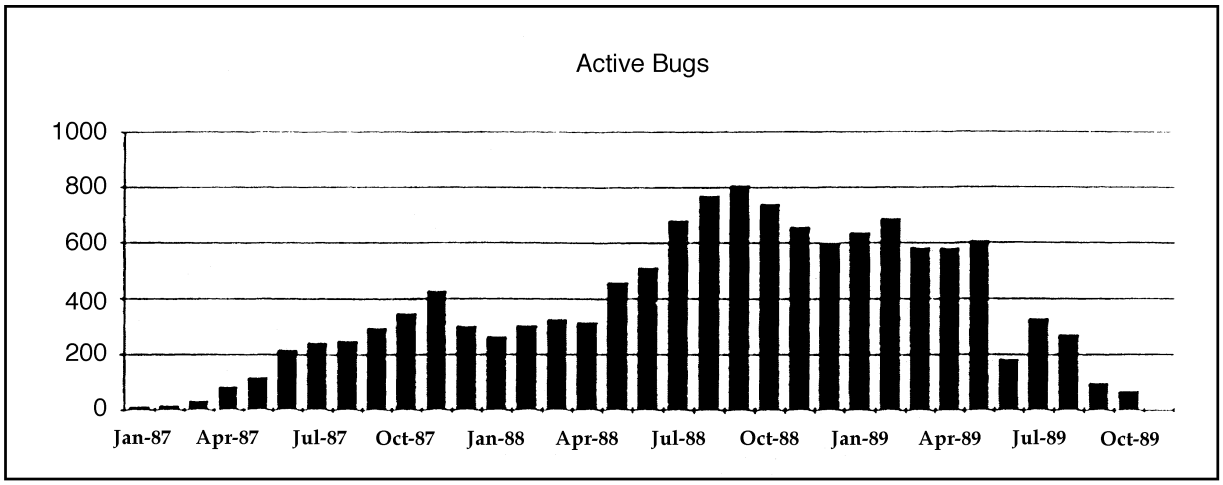
Exhibit 5 Data from Opus Portmortem

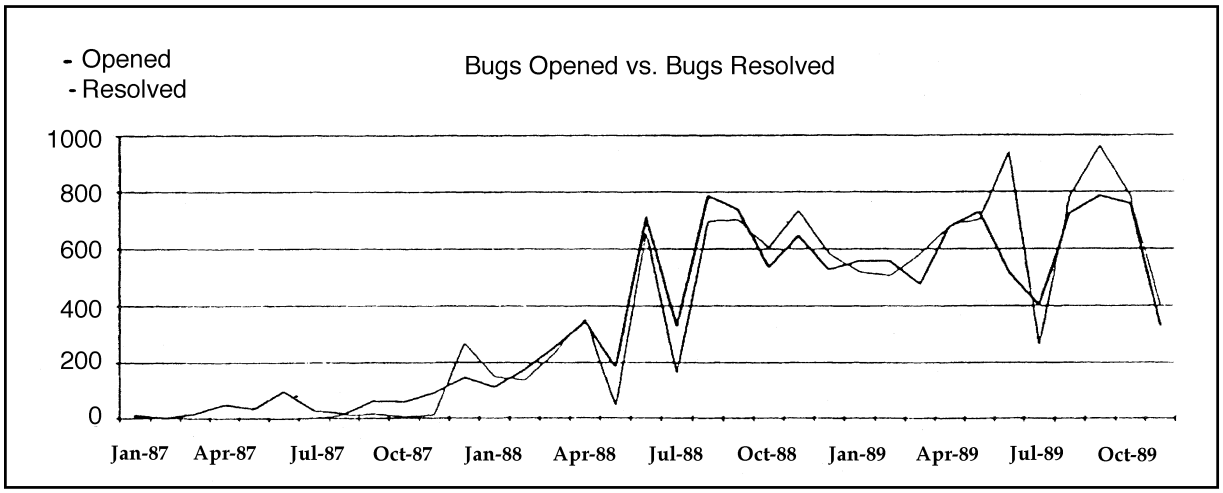
Statistics	
Size of shipped product	852,576 bytes[a]
Number of lines of program instructions (code)	249,000
Total development time spent (including part-time members	55 person-years
Number of lines of code present at "code complete"	209,000

One "byte" corresponds to a unit of computer memory storage roughly comparable to that required by one character of text.

asm = assembly language code.

Exhibit 5 (continued)





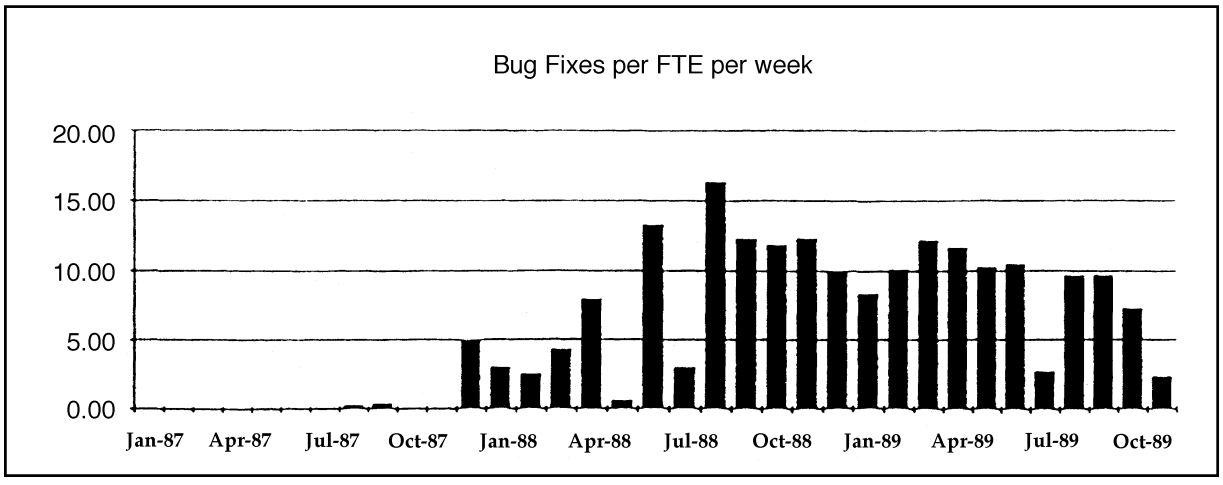
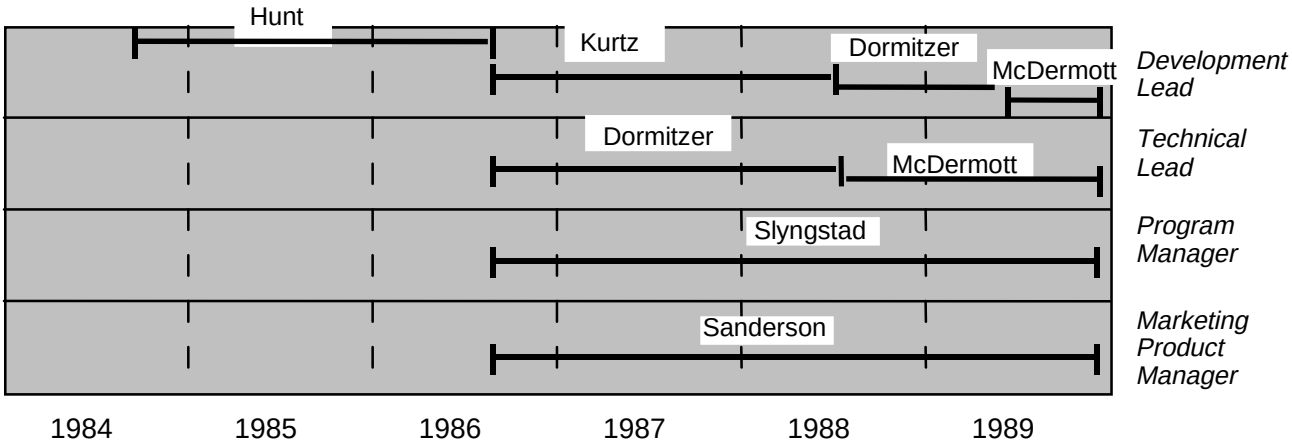


Exhibit 6 Time Line of the Cashmere/Opus Project

Cashmere/Opus Leadership



Cashmere/Opus Events

Date	Event
8/84	Winword project beings
8/85	Coding begins
9/85	First demonstration to Bill Gates
11/85	Coding work intensified
1/87	Testing work begins
4/87	Specification discussed with Bill Gates
6/87	Revised specification distributed
10/87	Addendum to final specification distributed
12/87	Visual freeze: the way the program looks to the user is officially established
2/88	Presentation to Bill Gates; additional features added
3/88	Addendum to final specification distributed
6/88	Almost all features reported to be "working to some degree"
7/88	Addendum to final specification distributed
8/88	Feature complete: no more features added after this point
10/88	Code complete
3/89	Performance optimization declared complete
10/89	Word for Windows is officially announced
11/89	Word for Windows ships

Exhibit 7 The PC Word Processing Software Market

Word processing software was the largest segment of the applications software market for IBM PCs and compatibles. Over $800 million was spent annually on professional word processors, the high end of the word processor market. This market segment was dominated by a few players, all of which were character-based in 1990.

WordPerfect, Version 5.1

WordPerfect version 5.1 was the latest in a long line of updates of a highly successful product. WordPerfect had the largest share of the word processor market at 45%. In product comparisons, WordPerfect generally won on the number of features it had and the wide variety of printers it supported. WordPerfect was also available on a wide variety of non-DOS computers including the Apple Macintosh, Digital Equipment Corporation's VAX line, and IBM mainframes. Although WordPerfect was character based, WordPerfect Corporation had announced that a Windows version was under development and was expected to ship in the first quarter of 1991.

Word for DOS, Version 5.0

Microsoft's original high-end, character-based wordprocessor for DOS had first been released in 1983, but had not received good reviews in industry magazines or achieved much market success. Since then, however, Microsoft had continually improved the product to the point where it was rated on a par with WordPerfect, and much higher than any other word processors. Word's market share had improved correspondingly and was now the second highest at 30%.

Wordstar, Version 5.5

Wordstar Corporation had first developed a word processor for the CP/M computer market (a predecessor of IBM/DOS). Dominant in the CP/M market, it had translated its software to run under DOS in 1982, and had quickly become the market leader. Since that time, however, its updates had not kept pace with the rest of the market and the product had fallen behind technically. Its market share in 1990 was 12%, well below the market leaders.

Displaywrite 4

IBM's entry in the word processing software market was generally considered to be an inferior product to either of the two market leaders. Its limited market success (10% share) was attributed to IBM's dominance in the hardware market. Its market share had been declining for several years.

Exhibit 8 Word Processor Operating Environments

	Macintosh	IBM PC and Compatibles			
Operating System	**Mac/OS**	**DOS**	**DOS**	**OS/2**	**OS/2**
Layered Product[a]	—	—	Windows	—	Presentation Manager
Interface Type	GUI[b]	Character	GUI	Character	GUI
MicrosoftWord Processor	Word for Macintosh	Word for DOS	Word for Windows	—	—
WYSIWYG[c]	yes	no	yes	—	—

[a]The layered product is an add-on to the basic operating systems which provides a more uniform and easy-to-use user interface. The Mac/OS was designed from the ground up to provide such capabilities, whereas DOS and OS/2 do not. For this reason, Microsoft has developed Windows for DOS and the Presentation Manager for OS/2 to provide similar capabilities.

[b]GUI stands for Graphical User Interface, where the user gives commands to the computer by using a mouse to point to icons on a graphics screen.

[c]WYSIWYG stands for What You See Is What You Get. It refers to applications where the screen display closely resembles printed output.