

**Software: A Historic View of its Development as a Product and
an Industry**

Subah A. AL-Zayani

CSIS 550

04/26/2001

Table of Contents

Introduction	3
History of Software Development	5
The Pioneering Era (1955-1965)	5
The Stabilizing Era (1965-1980)	9
The Micro Era (1980 to the Present)	14
Software as a Product and an Industry	15
Major Developments During the 1980s	23
The Personal Computer Revolution	24
The Workstation	25
The Rise of System Integration and Outsourcing	26
The Federal Government's Role in the Development of the Software Industry	28
Summary	28
Works Cited	31

Introduction

Throughout history, many misconceptions have occurred regarding the actual meaning of Software. Some people refer to software as a set of programs that tell the computer what tasks to perform. Basically, it is the actual computer program stored in a computer. Within the realm of this definition of software, people tend to have a common misconception of the quality of a software program “productivity is measured by how many lines of code are produced in unit time” (Freeman 5). This misconception may lead to the production of large amounts of code that cannot be integrated together to successfully accomplish the customer’s needs.

Software Perspectives: The System is the Message, by Peter Freeman, provided an interesting description of some characterizations that would enable people to better understand the term “software”.

The brain and soul of a computer, not just paint.

The embodiment of the functions of a system.

The captured knowledge about an application area.

The collection of all the programs and data that are necessary to make a computer a special-purpose machine designed for a particular application.

All of the information (documentation) produced during the development of software-intensive system (Freeman 5).

Other people argue that software includes the defined processes or sets of rules that lead to the development of an output from a given input procedure

for solving a mathematical problem. It also includes the actual computer program, the program description, and the supporting materials (Kutten 1-2).

However, Kutten, the author of *Computer Software*, divided software into two categories: system and application. On one hand, the system software tells the computer how to work, while on the other hand the application software is a program that tells the computer how to perform a specific user job (Kutten 1-3).

An interesting definition of software is that software can be viewed as knowledge. Freeman explains this point of view by saying; “Any program certainly contains a large amount of information about a process and the data that are relevant to carrying it out. Programs also contain structural knowledge that indicates the relationship between different processes (programs) and their associated data, knowledge about the relationships among classes of data, and other information as well” (Freeman 27). Thus, Freeman emphasized the importance of understanding software better in all its different forms.

Therefore, no matter how people view software, and no matter how different the points of view may be, it is essential to understand the importance of software in the computer industry. This research is dedicated to the “soul of the computer” that has long been neglected by researchers, authors, and scholars, the computer software. The research will analyze the history of software’s development through three distinct eras. It will also provide an overview of software’s development as an “off-the-shelf” product through an overview examination of the forces that lead to its “unbundling.” Finally, the research will

analyze software's development into a multi-billion industry that revolutionized the modern computer era.

History of Software Development

Many authors used different terminology in tracing the different time zones in which software progressed dramatically. For example, Kutten divided the software evolution into three distinct stages: The Mainframe stage, the Custom stage, and the Micro Stage (Kutten 1-7). On the other hand, Robert L. Glass, author of "In the Beginning: Recollections of Software Pioneers," divided the long history of software into three different periods: The Pioneering Era (1955 -1965), the Stabilizing Era (1965 -1980), and the Micro Era (1980 - the present time) (Glass 12). Both of the software's evolution perceptions were similar, but, I found Glass' Software Eras more appealing.

The Pioneering Era (1955 – 1965)

During this era, computers were merely used in organizations or for governmental purposes "In 1954, a group of experts estimated that only fifty U.S. companies would be able to use a computer" (Kutten 1-7). Glass reflected on his experiences during that era, "The computer – the only one – is nearby, in something we call 'the machine room'." (Kutten 13). Computers during that era were extremely large in size, and were stationed in a separate room where the

users usually had to sign-up for machine time and then run the job themselves when their turn came (Glass 13). Basically, during that era, to run a job meant that you had to put the punched cards that contained the program, its data, and its control information, into the machine's card reader and wait for the results to come back on the printer. Therefore, programmers had to run their jobs. During this era, many computers invented every year or two, and each computer was different than the one before it. Therefore, software personnel had to rewrite their programs to run on the new computers as each one came along. Software was considered a "mere accessory to hardware" during the Pioneering Era (Kutten 1-7). Computer consumers bought or leased a computer from IBM, Burrough's, Sperry Rand, or any other manufacturer, while the manufacturers had grouped the system and application software into the pricing structure of the hardware, whether the user wanted it or not (Kutten 1-7). Unlike today, there were no software companies that sold packaged software.

Also, computer hardware during this era was designed for optimal performance of either scientific or business tasks. On one hand, business computers functioned with variable word lengths and performed decimal arithmetic. On the other hand, scientific computers used fixed word lengths and performed binary arithmetic (Glass 15).

As new machines were developed rapidly, software personnel had to rewrite and translate old software to meet the needs of the new machines. Each new computer had its own languages. The process of rewriting old software led the software programmers to use the method of "automatic translation of one

assembly language to another” (Glass 15). Even though the computer often translated 60 to 80 percent of the code, the remaining difficult and complex part of the code required manual human translation. Programmers debated about the nature of their job and whether real programmers should use assembly languages or High Order Languages (HOL). Therefore, they found it difficult to keep up with the fast pace of new computer and assembly language inventions, “The pace of change was horrendous, far faster than at any time since” (Glass 15).

In order to increase the amount of software availability, manufacturers had come up with an interesting plan that encouraged the customers to band together into software sharing “banks” (Kutten 1-7). The main idea behind the “software banks” was that each customer would deposit its own software and could withdraw other software contributed by other customers. However, these “banks” did not succeed, as companies that developed better and more efficient programs had no desire to help their competitors (Kutten 1-7).

On the other hand, as the notion of reuse flourished, and because software was free, user organizations commonly gave it away. Therefore, different groups offered catalogs of available reusable components. For example, IBM’s scientific user group SHARE offered such catalogs that contained mostly mathematical routines, such as trigonometric functions (Glass 16). Glass recalled “To have some of your products in the SHARE catalog was the highest honor the field offered. I recall being deeply proud when a debugging routine I had written was listed with SHARE” (16).

Grace Hopper, in 1957, developed a new compiler called MATH-MATIC as a refinement of her earlier invention, the A-0 compiler. During the same year, Sperry Rand released this compiler commercially for its UNIVAC (www.computerhistory.org). These earlier works on the A-0 and A-2 eventually led to the development of the first English language business data processing compiler, the B-0 (FLOW-MATIC) (www.computerhistory.org). In 1963, the ASCII (American Standard Code for Information Interchange) permitted machines from different manufacturers to exchange data. The code consisted of 128 unique strings of ones and zeroes, and each sequence represented a letter of the English alphabet, an Arabic numeral, an assortment of punctuation marks and symbols, or functions such as a carriage return (www.computerhistory.org). This development was extremely important in relation to standardization within the computer industry.

A	1	0	0	0	0	0	1
B	1	0	0	0	0	1	0
C	1	0	0	0	0	1	1
D	1	0	0	0	1	0	0
E	1	0	0	0	1	0	1
F	1	0	0	0	1	1	0
G	1	0	0	0	1	1	1
H	1	0	0	1	0	0	0
I	1	0	0	1	0	0	1
J	1	0	0	1	0	1	0
K	1	0	0	1	0	1	1
L	1	0	0	1	1	0	0
M	1	0	0	1	1	0	1
N	1	0	0	1	1	1	0
O	1	0	0	1	1	1	1
P	1	0	1	0	0	0	0
Q	1	0	1	0	0	0	1
R	1	0	1	0	0	1	0
S	1	0	1	0	0	1	1
T	1	0	1	0	1	0	0
U	1	0	1	0	1	0	1
V	1	0	1	0	1	1	0
W	1	0	1	0	1	1	1
X	1	0	1	1	0	0	0
Y	1	0	1	1	0	0	1
Z	1	0	1	1	0	1	0

Fig 1.1 ACSII Code

Elmer C. Kubie published an article in the *Annals of History of Computing*, "Recollections of the First Software Company," in which he identified the Computer Usage Company (CUC) as the world's first computer software company. Kubie and John W. Sheldon were the founders of the Computer Usage Company in March 1955. An interesting incident occurred while IBM was working

on its System/360. While the presidential election was about to be held in 1964, IBM planned to use an early machine to handle data and produce reports for the CBS TV network (Kubie 68). IBM also decided to show its IBM System/360 to its TV audience, but the network insisted that any machine displayed had to process real data. Because there was no software produced to accomplish that purpose, CUC was asked to use software they had to develop and write a program that would keep track of the nation's reporting precincts (Kubie 68). Therefore, the IBM System/360 successfully processed real data that election evening.

The Stabilizing Era (1965 – 1980)

During this era, employees still did not have computers on their desks. Computers were still located in “machine rooms”. Unlike the Pioneering Era, programmers no longer had to run their jobs, because the “whole job-queue system had been institutionalized” (Glass 17). As



Fig 1.2 IBM System/ 360

Glass reminisced on his job experiences at Boeing, he explained how an “enormous bureaucracy” had grown around the central computer center. He called it a “bureaucracy” because there was much subdivision managed by sets of appointed officials who followed an extremely inflexible routine. There were people who were assigned to transport jobs between pickup locations scattered throughout the company and the computer

center. There were also other people who scheduled the central computer's time, and decided what ran and when. Others operated the computers and controlled the computing output, so that the printouts would not fall in the wrong hands (Glass 17).

When IBM released its System/360 mainframe in 1964, it was considered as "probably the most significant event in the history of operating systems" (Deitel 12). The IBM System/360 was designed with electronic circuits that permit nanosecond bit manipulation speeds (Bradley 3). With the development of the IBM System/360, customers no longer had to change computers every few years, and programmers no longer had to rewrite the old software every few years as well. Software programmers finally got the opportunity to write new software and "diminish the time spent on rewriting the old" (Glass 18). The IBM System/360 offered both decimal and binary arithmetic, thus the machine had the ability to perform both scientific and business tasks. Of course a high-tech computer such as the IBM System/360, for its time, needed a massive operating system. The operating systems at that time were still distributed free with computers by the manufacturers. Each System/360 was promised a full OS360 support program. IBM recognized the problems of slow, awkward, and incompatible conversion between systems, and designed the System/360 series to be "architecturally compatible, to use the same operating system (OS/360), and to offer greater computer power as the user moved upward in the series" (Deitel 12). One of the System/360's features was the JCL (Job Control Language). Instead of using the old-fashioned punched-cards where

programmers simply had to stuff the cards into the card reader and press start, they now had to write a program in a whole new language that told the computer and OS360 what to do. Another interesting feature in the System/360 was the emulator, which was a microprogrammed feature that allowed the System/360 to emulate the instruction sets of earlier systems (Glass 37). Therefore, the early users of IBM 7000 and 1400 systems could continue to run their old applications on the System/360 machines without the new System/360 programming support. IBM provided four main software support areas for the System/360:

1. Model 20 Programming Support.
2. Basic Programming Support.
3. Basic Operating System (Tape Operating System, Disk Operating System).
4. Operating System (Bradley 16).

Because of the successful merger of both the scientific and the business applications in the IBM System/360 and its operating system, IBM wanted to expand its merging intentions by merging all programs onto one language. Therefore, IBM borrowed features from other languages such as FORTRAN (Formula Translator), created in 1954, COBOL (Common Business Oriented Language), created in 1959, ALGOL (Algorithmic Language), and LISP in order to create the PL/1 (Tremblay, Bunt, and Richardson 2). As a result, the PL/1 became a very large and general language, with features appropriate for different problems.

During this era, programmers were able to concentrate on building new products as the software field stabilized. With the stabilization of software came major changes, for software became a corporate asset and its value increased as years passed by. Also, software maintenance grew as a profession “perhaps the largest growth profession of the era” (Glass 20). Software maintenance was not solely related to correcting errors; in fact it was more connected to software enhancement than error correction.

A new concept appeared during this era, “structured programming,” that had a major influence on the programmers’ techniques of coding. This concept was introduced programmers wanted to develop more efficient programs with fewer lines of code. Structured Programming was defined, as “the task of organizing one’s thought in a way that leads, in a reasonable time, to an understandable expression of a computing task” (Basili and Baker 3). The major objective of Structured Programming was to develop methods that would simplify the program development process. By using a few simple structures that aid in minimizing the number of interactions and interconnections of which a programmer had to be aware of, and by keeping program segments manageable through minimizing their size, a problem becomes broken up into small and easily understood pieces (Basili and Baker 5).

New developments in operating systems occurred during the Stabilizing Era. In 1969, two programmers from AT&T Bell Laboratories, Kenneth Thompson and Dennis Ritchie developed the UNIX operating system. The duo wanted to create a computing environment for facilitating programming research

and development. The operating system was designed to be to be a convenient system for supporting program development (Deitel 481). UNIX offered many features, but most importantly timesharing and file management (www.computerhistory.org). By 1975, UNIX had become very popular in many universities nationwide, and the first release of UNIX for public consumption was version five (Deitel 481). Also, in 1976 Gary Kildall developed another operating system for personal computers, the CP/M. CP/M is an abbreviation for “Control Program for Microprocessors”. CP/M was an operating system that allowed the user to take full advantage of the computer hardware (Weber 14). This operating system made was a group of programs stored on a diskette known as the “system diskette” (Weber 14). CP/M was loaded from the diskette whenever the system was properly turned on, and it began monitoring the keyboard for commands. The operator then entered the appropriate command to activate the desired program. Once the program had been loaded, the operator could use it, and after the program ends, the CP/M prompt would appear again and await the entry of new commands via the keyboard (Weber 15). Basically, the CP/M made it possible for one version of a program to run on a variety of computers. CP/M became extremely accepted among personal computer users, and it evolved through a series of increasingly powerful versions (Deitel 538).

The Micro Era (1980 To the Present)

The last and current era began with the rise of microcomputers. In the past, programmers had to take their jobs to a terminal or batch pickup points, but during this era they had the computer power within easy grasp. Glass explains how microcomputers positively affected his job “I no longer need to multiprogram, working on more than one problem at once, wasting time when I ‘context switch’”(Glass 22).

A major breakthrough in computing during the 1980s was the development of Graphical User Interface (GUI) that was both user-friendly and programmer-friendly. GUI is an application environment that has the ability to work with graphical objects (Powell 355). One benefit of GUI is the users’ ability to learn a second application faster because they are not familiar with the environment. It also allows the user to see the final product before he or she prints it “what you see is what you get” (Powell 357). Another breakthrough during this era was the status of software as a commercial product with its own industry. In this era, software was no longer free, for the value of software was firmly established, and software companies joined many of the hardware in the Fortune 500.

The Microsoft Disk Operating System (MS-DOS) made its debut as the basic software for the released IBM PC in 1981. The following year, Mitch Kapor developed the Lotus 1-2-3. He wrote the software directly into the vide system of the IBM PC, thus it bypassed DOS and ran much faster than its competitors

(www.computerhistory.org). In 1983, Microsoft announced it had created a new word processor formerly named “Multi-Tool Word”, now known as “Word”. It also announced the creation of Microsoft Windows, the user-friendly software that eventually changed the software industry. To market the new product, Microsoft distributed 450,000 disks demonstrating its Word program in the November issue of *PC World* magazine (www.computerhistory.org). The first successful version of Windows that had finally been able to satisfy PC users was Windows 3.0, shipped by Microsoft on May 22nd, 1990. The application’s interface was changed, for Microsoft created a design that allowed PCs to support large graphical applications and run multiple programs simultaneously (www.computerhistory.org).

Software as a Product and an Industry

In the early 1960s, the U.S government contracted with independent software firms to develop software for defense and space projects. Due to the delicate nature of the projects and their complexity, the programs had to meet strict reliability and real-time constraints (Kutten 1-8). Therefore, the contractors developed efficient software production methodologies that eventually “spun-off” into the commercial market for commercial program development (Kutten 1-8). The first software programs were customized programs that fulfilled unique customer needs. As software development expanded, it was still held back by a major constraint: manufacturers bundled software with hardware. Therefore, no

matter how good software was, customers would not pay for it when they received equivalent software for free from the hardware manufacturers. Luanne Johnson, author of the article “A View From the 1960s: How the Software Industry Began,” published in the *Annals of History of Computing* magazine, re-emphasized this point by saying: “Software was either given away free by the computer manufacturers or written specifically for each computer installation,” she continued by stating that “One cannot make any money selling software” (Johnson 36). However, Luanne Johnson made these statements with regards to the software’s status in the 1960s. Yet the market developed and grew, and the companies that used to develop customized software began to develop standardized programs for mainframe and mini-computers that could be marketed diverse companies in different industries. By 1965, there were about 45 independent software suppliers, and the first internationally marketed program, “Autoflow,” was sold (Kutten 1-8).

During the early years of the Stabilizing Era (1965-1980), a few daring software houses began to offer products, and software began to have value. However, it was still a difficult transition, for the major hardware companies intentionally kept their software or the price of the software low, to prevent the growth of a separate software industry (Glass 19). In 1967, the first million-dollar software product, the Mark IV, was introduced by Informatics (Johnson 39). The Mark IV File Management System became one of the most successful software products ever sold in the Stabilizing Era. Mark IV was conceived as “a non-application-oriented software product, the first true such software in the field”

(Postley 48). It was designed to run on all IBM System/360 machines. Mark IV was also the considered to be the first software product to reach cumulative sales of \$10 million and then of \$100 million (Johnson 41). By 1979, there were more than 4,000 installations of Mark IV in 45 different countries and more than half of the Fortune 500 companies. At that time, the top ten Fortune 500 companies used IBM equipment, and all of them were Mark IV customers (Postley 50).

In December 1968, IBM announced that it was going to change its pricing policies on software by establishing a task force that would review the current policies and recommend a new one. The new policies were to have been announced in June 1969. The new IBM pricing policies was a hopeless attempt to prevent a governmental antitrust suit (Johnson 41). Attorney General Ramsey Clark directed the Justice Department to file an antitrust suit against IBM. The Justice Department claimed that IBM had gotten too large and was “manipulating the field” (Glass 21). Among the government’s allegations were:

1. IBM’s policy of maintaining a single price for its hardware, software, and related products forced customers to buy only IBM products.
2. IBM used its accumulated software and related products to preclude competitors from competing competitively. (Kutten 1-8)

Although IBM denied all charges, it unbundled its software as a partial response. On June 30, 1969, IBM announced that it would reduce the price of its

hardware leases by 3 percent and began to charge separately for customer training, for system engineering services, and for some software beginning on the 1st of January 1970 (Johnson 41).

Therefore, IBM created history when it announced on June 23rd 1969, that it would begin pricing software separately from hardware beginning on January 1st, 1970. An article in the *Computerworld* magazine, “Software Industry Born With IBM’s Unbundling”, explained how IBM’s unbundling of software and hardware was a great help to the growing of the software industry (qtd. in Johnson 36). In addition, the growth of independent software vendors made it possible for IBM to consider separate pricing for software and to retreat from its commitment to provide all of the software tools that users might have needed in order to purchase or lease IBM computers (Mowery 25).

An interesting story was told by Watts Humphrey, who was a software pioneer at IBM, about software being marketed as a product without being attached to hardware:

Humphrey recalls, “IBM San Jose introduced a new file product.”

When it was introduced it did not sell well. “The product developers found that IBM’s old sort program had not been upgraded for the new file product,” and for the first time, Humphrey recalls, “hardware managers found that, without programming support, hardware would not sell” (qtd. in Glass 28).

Because hardware manufacturers still thought that their job was selling hardware and not software, they left the market open for suppliers who had no connections with the manufacturers to enter the marketplace and satisfy the customer demand. Because software had become unbundled from the price of hardware, software was gradually being seen as a product. The explosive growth of the personal computer field also made it “abundantly clear to most people that software can indeed be treated as a product” (Freeman 25). Companies emerged that were solely devoted to the creation of software and selling it as a product to different users. Larry Welke, author of “How the ICP Directory Began,” stated that they had estimated that “in 1969 about 9% of the programs [they] listed were developed specifically as software products. By 1973, that number was up to 49%” (Welke 2). Interestingly enough, many of the software millionaires are not “old-line” professional programmers, but rather people who quickly grasped the need for a particular software product. Programmers like Bill Gates, who had vision for a future and the creation of a software industry significantly contributed to the rise of software as a product. In an interview, Bill Gates once said “We’re going to create the software that puts a computer on every desk and in every home” (Lammers 82). People like Bill Gates created the software product, packaged it, and managed to get into the market first. Therefore, they generated millions and millions of dollars in revenues. After the different software programs were created and sold to a diverse customer base for different causes and satisfy different needs, it was logical to conclude that “all software is ultimately a product for someone else” (Freeman 26). Software had

established a firm status as a product with a separate industry than that of the hardware. Unfortunately, the software industry had to struggle in its early ages of development. Luanne Johnson, the author of “From Not-Invented-Here to Off-The-Shelf,” explained the difficulty of convincing the customers to purchase the off-the-shelf software product. In her article, she said “the sales process with every one of my prospects consisted of trying to convince them that my off-the-shelf software would be able to do the job for them without requiring a lot of modification to either the software or their business practices” (Johnson 1).

As more and more software vendors appeared, most hardware vendors retreated from the software production. On the other hand, recent entrants into the computer production industry took a minor participation role in software production. The software industry had benefited from the prior existence of an enormous number of small contract programming companies that originated from the needs of larger companies and the government for custom software. These small companies led to a strong infrastructure on which the software industry expanded. During the period between 1965 and 1970, the new entrants that formed the base of the software industry included software tool and utility program suppliers, as well as companies that provided applications for particular industries and for common software needs. Therefore, the period between 1965 and 1970 was crucial period that lead to the emergence of the current structure of the U.S software industry (Mowery 24).

As shown in Table 1.1 below, there was a significant increase in the number of companies in the software industry. From 1975 to 1981, there were a total of 3,919 companies in the software industry. However, in 1982, there was a significant increase of approximately 400 new companies, totaling 4,340 companies. These figures include all software product, professional services, and integrated systems companies. On the other hand, table 1.2 indicates the number of employees in the software industry within the 1979-1982 periods. As shown in the table, between 1979 and 1982, the U.S. software industry employment in software products and profession services grew from 65,400 to 178,000 employees.

Table 1.1 Number of Companies in U.S. Software Industry (Kutten 1-10).

Type of Company	1979	1980	1981	1982	Net Gain (79-82)
Software Products	1,095	1,225	1,605	1,879	+78.4
Professional Services*	820	978	1,284	1,348	+52.8
Integrated Systems	N.A.	N.A.	1,030	1,113	N.A.
Total U.S. Software Industry	N.A.	N.A.	3,919	4,340	N.A.

* Includes contract programming, systems, management and design, and computer related consulting, education, and training.

Table 1.2 U.S. Software Industry Employment by Sector 1979-1982 (Kutten 1-10).

Sector	1979	1980	1981	1982	Annual % Change (79-82)
Software Products	22,400	40,100	51,100	68,000	+45
Professional Services (1)	43,000	51,000	103,000	110,000	+37
Subtotal	65,400	128,100	154,000	178,000	+40
Integrated Systems (2)	N.A.	34,700	43,000	46,000	N.A.
Total (3)	N.A.	162,800	197,000	224,000	N.A.

(1) Includes contract programming, systems management and design, and computer related consulting, education, and training.

(2) Data unavailable before 1980.

(3) Excludes self-employed individuals.

As the microcomputer software market rapidly grew and changed. The microcomputer software market reached an estimate of more than 5000 active software companies by 1987 (Kutten 6-2). Programmers and companies knew that creating a popular program could yield extremely high revenues. Bill Gates and his Microsoft dream, for example, made him the richest man in the U.S, and possibly in the world. Up until 1980, large microcomputer software publishers did not exist; this was about to change. During 1981, large book publishing houses, such as McGraw Hill, Prentice Hall, and John Wiley & Sons entered the market (Kutten 6-2). A product that was once unknown, now had its own stores; software-only stores had appeared during that period. Even large bookstore chains began to sell software.

Major Developments During the 1980s

The early 1980s marked the period of expensive software. In the past, a programmer or a group of programmers, with the hope to generate mild profits, produced software. However, with the increase of the software industry, the cost to develop and market the products became expensive. A developer now had to “proclaim to the world why his program is better than that of others” (Kutten 6-4). Before the 1980s, the average software user was a person who had knowledge about how the software worked. However, during the 1980s the software market drastically expanded, average software users needed detailed documentation, user-friendly instruction manuals, training guides, and demonstration diskettes, which programmers did not have the ability to produce (Kutten 6-3)

In 1987, the receipts of U.S software programming service companies were \$14.2 billion, the receipts for computer integrated systems design were \$7.1 billion, and the receipts from prepackaged software sales were \$5.9 billion (Mowery 18). The first two forms of revenues were derived from the production of custom software for specific companies. On the other hand, packaged software was considered to be an intermediate good that provided applications solutions for large numbers of users (Mowery 19).

Also, during this period, three major developments affected the software industry. At the beginning of the decade, the emergence of the personal computer provided a new and revolutionary organizing principle, mass publication of packaged software (Mowery 31). Another major development was

the introduction of the workstation in the middle of the 1980s that “fueled the creation of a large new class of software applications that exploited the workstation’s sophisticated graphics and numerical computation capabilities” (Mowery 34). Finally, the rapid growth of system integration companies and “outsourcing” of corporate data processing activities and management that was supported by the productivity and organizational problems that occurred in the 1970s.

The Personal Computer Revolution

IBM introduced its personal computer (PC) in August 1981. The personal computer was a machine that combined a reasonable level of computational power and an operation system that facilitated application development (Mowery 34). The microcomputer market swiftly grew during the 1980s. The rapid growth of the microcomputers in turn provided a standard market for operating systems and application software. The tremendous size of the personal computer market opened new doors of opportunity for software producers to create and sell large numbers of customized software products. These profit opportunities were simply unavailable in the markets for larger computers, “even though software in the latter markets often had higher price tags” (Mowery 34).

With the growth of the personal computer market, many new software companies emerged. Lotus, in 1985, recorded annual revenues of \$226 million, and had become the sixtieth largest U.S. data processing company (qtd. in

Mowery 34). Microsoft reached \$163 million of revenues, and ranked seventy-eight, while Ashton-Tale's revenues recorded \$110 million and ranked one hundredth. Later in 1988, WordPerfect Corporation reached \$179 million in revenues. A single product penetrated a large share of the installed base of personal computer and dominated a class of software. Lotus had the spreadsheets, Microsoft had the operating systems, Ashton-Tale had its databases, and WordPerfect Corporation had the word processing (Mowery 35).

The market in personal computer software resembled that of the publishing and mass entertainment. Promotion and distribution methods used in marketing the software products were similar to that of the book and music recording industries. Personal computer magazines that prospered during that era and included vast advertising for software and hardware strengthened the similarities between the new software industry and the recording and publishing industries.

The Workstation

The workstation was introduced by Apollo in 1981 and Sun Microsystems in 1982, and was a crossbred of the personal computer and minicomputer (Mowery 39). The workstation took advantage of reductions in the price of microprocessor circuits. Workstation producers entered the market and sought to attract engineers and other technically sophisticated users who would contrarily use minicomputers or mainframe computers. The major appeal of the workstation

was its “graphic capability” (Mowery 39). The graphic-intensive applications involved computer-aided design (CAD) and computer-aided engineering (CAE) that had been developed for minicomputers. However, these applications suffered from limitations in graphic display capabilities and the fragmentation in the minicomputer software market. Sun Microsystems adopted a corporate strategy that involved the use of Unix. The strategy was to influence technical users and software developers that applications for its workstation would be transferable to more powerful workstation products. In the period 1985 to 1990, “the number of suppliers listed in Sun Microsystems’ Catalyst guide to software for Sun workstations grew from 177 to 1,325” (Mowery 39). The majority of companies offered a single software product that was often a specialized solution to a problem.

The Rise of System Integration and Outsourcing

During the 1980s, personal computers did not have the performance or capacity to replace mainframe computers. The organization also faced problems with maintaining and expanding mainframe software applications. The continued growth in the intensity of computer operations during the 1980s provided an increasing challenge for internal development capabilities, who needed to provide solutions to problems that required sophisticated corporate business operations (Mowery 40). These internal capabilities drained the companies’ resources and distracted their necessary business activities. Therefore, firms

began to reconsider the make-or-buy decision for their entire data processing activity. It was cost-beneficial to search for firms that could provide technological knowledge and human resources to implement specialized software solutions, choose among complex competing hardware offerings, and deliver useful information services to internal users. The growing complexity of data processing technologies and markets caused the companies to purchase software solutions instead of developing them, which in turn led to the birth of new companies that satisfied this rising demand.

The 1980s proved to be a complex period in the development of the software industry. Mainframe and minicomputer applications and sales grew during this period, yet the widespread adoption of workstations and personal computers led to complexities. Personal computer software companies took a “publishing” approach that helped them achieve economies of scale in software development. The large installed base of workstations and the use of Unix as a common operating system supported the development of specialized, computationally intensive software (Mowery 41). With large-scale organizations, effective distribution methods, and a stable number of users, the software became a mature industry.

The Federal Government's Role in the Development of the Software

Industry

The federal government's policy during the postwar period highly influenced the computer software industry. The software industry received considerable federal research and development funding throughout the postwar for cold war defense, especially strategic air defense. The funding of the defense-related software had created an infrastructure that supported a new area of R&D, training, and technology development (Mowery 53). During the early years of the postwar period, private industries had been responsible for a great deal of innovation in software. By the 1960s, the industrial innovations drew on research and manpower that had been generously supported by federal government funds. Direct "spillovers" from defense-related support appeared. These "spillovers" were widely adopted civilian versions of software developed initially for military applications. Advances from the private sector supplemented them (qtd. in Mowery 54). Universities also received federal funding and developed several important programming languages and operating systems.

The Department of Defense had growing concerns regarding the soaring costs of projects, delays, and unreliability of complex, software-intensive weapons systems. The Strategic Defense Initiative of the 1980s was aimed to reduce the cost of defense-related software by utilizing commercial software for military applications (Mowery 54). The federal R&D also supported the creation of computer science as a new academic discipline.

Summary

In the past fifty years, different innovations in semiconductors, data storage devices, computer architecture, software, and data communications had revolutionized information collection, storage, processing, and distribution (Mowery 15). However, the role of computer software in these developments had been neglected. Every application of information technology had required complimentary software.

The explosion of computing power and dramatic reductions in hardware costs that had resulted from the development of microprocessors had an interesting implication on the role of users in the development of the traded software industry. The large size of the U.S. packaged software market had given the U.S. firms that pioneered in their domestic packaged market a formidable “first-mover” advantage that later became exploited internationally (Mowery 7).

Although critics argue that the software industry had found its roots long before IBM’s “unbundling,” it was the antitrust case that officially caused the creation of a completely new software market. The IBM’s unbundling helped to legitimize the concept of paying for software.

This research was dedicated to the soul of the computer, software, which had long been neglected by researchers, authors, and scholars. The research analyzed the history of software’s development through three distinct eras, the Pioneering Era, Stabilizing Era, and Micro Era. It also provided an overview of

software's development as an "off-the-shelf" product through an overview examination of the forces that led to its "unbundling." Finally, the research analyzed software's development into a multi-billion industry that revolutionized the modern computer era.

Works Cited

- Basili, Victor R., and Terry Baker. *IEEE 1975 Structured Programming*. IEEE Computer Society, 1975.
- Bradley, John H. *Programmer's Guide to the IBM System/360*. McGraw-Hill Inc., 1969.
- Deitel, H.M. *An Introduction to Operating System*. Addison-Wesley Publishing Co., 1983.
- Freeman, Peter. *Software Perspectives: The System is the Message*. Addison-Wesley Publishing Co., 1987.
- Glass, Robert L. *In the Beginning: Personal Recollections of Software Pioneers*. IEEE Computer Society Press, 1998.
- Johnson, Luanne James. "A View From the 1960s: How the Software Industry Began." IEEE Annals of the History of Computing. Vol. 20, No.1, 1998.
- ---. "From Not-Invented-Here to Off-The-Shelf." The Software History Center. 1997: 2 p. Internet. 12 Apr. 2001. Available at: <http://www.softwarehistory.org/Johnson2.htm>
- Kubie, Elmer C. "Recollections of the First Software Company." IEEE Annals of the History of Computing. Vol. 16, No. 2, 1994.
- Kutten, L.J. *Computer Software: Protection / Liability / Law / Forms*. New York: Clark Broadman Co. Ltd., 1988.

- Lammers, Susan. *Programmers at Work: Interviews with 19 Programmers Who Shaped the Computer Industry*. Microsoft Press, 1989.
- Mowery, David C., *The International Computer Software Industry: A Comparative Study of Industry Evolution and Structure*. New York: Oxford University Press Inc., 1996.
- Postley, John A. "Mark IV: Evolution of the Software Product, a Memoir" IEEE Annals of the History of Computing. Vol. 20, No. 1, 1998.
- Powell, James E. *Designing User Interfaces*. Microtrend Books, 1990.
- Tremblay, Jean-Paul, Richard B. Bunt, and Judith A. Richardson. *Structured PL/1 (PL/C) Programming*. McGraw-Hill Inc., 1980.
- Weber, Jeffrey R. *CP/M Simplified*. Weber Systems Inc., 1982.
- Welke, Larry. "Who the ICP Directory Began." The Software History Center. 1998: 2 p. Internet. 12 Apr. 2001. Available at: <http://www.softwarehistory.org/Welke1.htm>
- (www.computerhistory.org) Computer History. "Timeline Software Page." <http://www.computerhistory.org/timeline/topics/software.page> (8 April 2001).
- *Fig 1.1 ACSII Code*. Computer History. "Timeline Software Page." <http://www.computerhistory.org/timeline/topics/software.page> (8 April 2001).
- *Fig 1.2 IBM System/360*. Computer History. "Timeline Software Page." <http://www.computerhistory.org/timeline/topics/software.page> (8 April 2001).