The U.S. Software Industry:

An Analysis and Interpretive History

W. Edward Steinmueller

Professor of the Economics of Technical Change

Maastricht Economic Research Institute in Innovation and Technology

University of Limburg

The Netherlands

March 14, 1995

Prepared for University of California, Berkeley

International Computer Software Industry Project

Directed by David C. Mowery

Forthcoming in David C. Mowery (ed.), "The International Computer Software Industry," Oxford University Press, 1995.

Acknowledgements: The author is grateful to the Markle Foundation and the Center for Economic Policy Research for financial support underlying this research, to David Mowery for his substantial intellectual and editorial contributions to this chapter, and to others who have helped me develop the ideas and expression of this chapter including Timothy Bresnahan, Franco Malerba, Tom Cottrell, Salvatore Torrisi, Yasunori Baba, Peter Grindley, and Robert Merges.

I. Introduction and Economic Foundations

During the past fifty years innovations in semiconductors, data storage devices, computer architecture, software, and data communications have revolutionized information collection, storage, processing, and distribution, creating new industries and transforming industries inherited from past industrial eras. Explanations of this revolution in information technology have focussed on the extraordinary reductions in the cost of the hardware components,[1] largely to the neglect of the role of computer software in these developments. Nonetheless, every application of information technology has required complementary "software"--computer instructions that transform the *tabula rasa* of computer hardware into machines that perform useful functions.[2]

This chapter offers answers for several basic questions about the historical development of the U.S. software industry. First, what determines the division of labor in software production among hardware producers, computer users, and the companies that produce software as their primary business, the "independent software vendors?" Second, have key events in the history of the U.S. software industry created a distinctly "American" system of software production that need not or cannot serve as a model for the development of the software industries in other nations? Third, what economic effects follow from the high fixed costs of initial software creation and the low marginal costs of reproducing software? In answering these questions this chapter will draw upon economic theory as well as descriptive material from industrial and technological commentaries.

Several definitions and distinctions are useful in setting the stage for answering our main questions. In this chapter, software is a general term of reference for instructions controlling the operation of information technology hardware. Programmers are people that devise specific

---

[1] For example, Gordon [1990] estimates that the annual rate of decline reduction in computer hardware costs averaged 19% for the from 1954-84 period.

[2] There is no clear boundary between hardware and software. Any information processing operation that can be achieved with "instructions" can also be achieved by an hardware subsystem. For example, systems may be designed that use software instructions for finding the square root of a number or, alternatively, an electronic component can be constructed that performs the same function within the system. Similarly, many electronic systems employ "programmable" components where a single set of instructions is permanently built in at the time of system manufacture. The software in these systems is "embedded" in the electronic system. The economics and industrial structure implications of embedded software, while of growing importance, are not considered in this chapter.

collections of instructions called software programs or, simply, programs. The use of the word "systems" is ubiquitous and unavoidable; it will be used here to refer to either complementary combinations of hardware and software or to collections of software programs that are "inter-operable," i.e. programs that operate together and exchange information. The technologies for acquiring, storing, processing, and transmitting information are collectively referred to as "information technology"and include both hardware and software components.

Software is classified as a "business service" in the U.S. income and product accounts despite the fact that "packaged software" products more closely resemble personal computer peripherals or books in their methods of distribution and reproduction. Software that is delivered as part of a business service can and should be distinguished from software that is sold as a product. Software services are performed within the programming services (SIC 7371) and integrated computer systems (SIC 7373) industries, although the latter industry performs hardware engineering design activities in addition to creating software. Sales of software as a distinct product are recorded as output of the packaged software industry (SIC 7372). Although the production of software is a labor-intensive activity, in principle software is reproducible at very low costs relative to the costs of its creation, a characteristic that is unusual in the service industries. This means that companies engaged in selling programming services will seek opportunities to reuse part or all of the software previously created for previous clients in selling service to new clients.[3] A particular software program that is only produced once should be viewed as a service output, while a program that is reproduced dozens or millions of times has development and marketing characteristics closer to those of manufactured goods. Understandably, such distinctions are not made in assembling statistics in U.S. income and product accounts, and the output of the packaged software (SIC 7372), programming services (SIC 7371) and integrated computer system design (SIC 7373) sectors contain an unmeasured mixture of "one-off," "reused," and reproduced software in each sub-sector.[4] Of course, creation of software for extensive reproduction is a primary aim in the packaged software (SIC 7372) sub-sector.

---

[3] The ability to reuse may, on occasion, be contractually limited.

[4] The term "reused" refers to the use of large blocks of instructions in the delivery of multiple outputs. Reuse allows software producers to "customize" software for clients more economically than designing from scratch. While reuse is assumed to be quite frequent, there is little evidence about its actual extent.

A second important distinction that is relevant in examining the software industry is the division of output between intermediate and final goods. Intermediate goods are used to produce other goods and services, while final goods are sold to consumers. With the recent, and nearly simultaneous, arrival of home computers, information services, and programmable consumer electronics systems (e.g. video game systems), independent software producers and system producers' sale of software as a final good have become a major industry. Indeed, the conceptual and technological distinctions between such "consumer" software products as games, and those historically considered to be "entertainment" such as music or video products, are rapidly fading as suggested by the increasing frequency of the term "multimedia" to refer to hybrids of video, sound, and software. Nonetheless, most of our attention in this chapter, as in the other chapters in this volume, is devoted to software that is an intermediate good, employed by businesses in the production of other goods and services or sold for the same purposes to other enterprises. We will ignore many of the problems of producing and marketing consumer-oriented software.

A third important distinction is between software and the production of other economic commodities. On the one hand, software, like other commodities, requires inputs that have alternative uses and, once produced, has economic value as an intermediate or final good. But, software is also an unusual economic commodity because its marginal costs of reproduction are very low or negligible. The low costs of software reproduction imply that society must grant businesses some right to control reproduction (and charge higher prices than the cost of reproduction) if investments are to be made in software creation, especially packaged software. Otherwise, third parties would make a business of reproducing software, and competition would drive the costs of software to the low marginal cost of its reproduction.[5] As a result, intellectual property protection is a key policy influence on firm strategy and evolution in this industry, as Chapter 10 points out, and the respective political influence of hardware manufacturers, custom software and service providers, and producers of packaged software may influence the structure of software-related intellectual property protection.

Computer producers, users, and independent software vendors each have distinct incentives to produce software. Computer system producers have incentives to produce software because software is an economic complement to the sale of computer systems, i.e. the availability

[5] This appears has happened in nations with no effective intellectual property right protection for software. At the same time, the rents available from protection are limited by the competition of "non-infringing" substitutes.

of software will increase the sale of computers. Ideally, computer producers can receive revenues from both the sale of computers and software. In practice, hardware producers in the U.S., with the notable exception of IBM, have received a diminishing share of their revenues from software production.[6] U.S. hardware producers involvement in software creation has followed a pattern of vertical "dis-integration'," favoring user-produced software and the entry of independent software producers.

Despite the incentives to combine hardware and software production, the ability of computer producers to understand and solve specific user problems is limited. The presence in the U.S. of an enormous variety of industries that use information technology has been a stimulus to the creation of a correspondingly large volume of user-created software. However, as in other nations, users have little incentive to sell the software they develop to rivals.

The potential profits from widespread sales of particular software products[7] have encouraged the entry of a third group of producers, "independent software vendors," (ISVs). The boundary in software producing activities among ISVs, computer producers, and users is determined by limits in the abilities of computer producers' ability to increase their "span of control" of joint hardware and software creation and by limits in the capabilities and incentives of users to produce software for sale to other firms. These factors, along with the design and marketing abilities of ISV firms, constitute the "infrastructure" of software creation in any particular nation and differ across nations with different patterns of industrial development. For example, the diversity of user-industries in the U.S. has made it very difficult for computer manufacturers to pursue vertical market strategies similar to those pursued by some European

[6] Moreover, although all software is complementary in demand with hardware, some software may raise the level of hardware demand more than others (e.g. software that demands intensive use of the computational or mass storage hardware), and we expect to find hardware producers more active in these areas than in other areas (e.g. in areas that permit users to conserve on their use of hardware).

[7] Information goods that can be protected from copying, can command prices in excess of the marginal costs of reproduction. If a given product's price provides more than sufficient revenues to recover the costs of developing, marketing, and maintaining the product, the producer will earn economic profits. Entry of independent producers is therefore likely if such entrants can overcome advantages of computer producers in the simultaneous design of hardware and software and computer users are disinclined or relatively inefficient at offering their internally produced software to others. As we will see, all of these conditions eventually were fulfilled.

computer manufacturers.[8] This was not because such strategies were unavailable, but because computer manufacturers' gains from a more exclusive pursuit of hardware improvement (combined with the vertical "dis-integration" of software production) were greater than their gains from controlling integrated software and hardware development in a large number of vertical markets. For example, Digital Equipment Corporation, abandoned its early position as a leading producer of integrated hardware and software systems for newspaper publishing, ceding the hardware integration and software development of such systems to other companies. In the U.S. market, most hardware vendors have retreated from software production, and as described below, recent entrants into computer production are minor participants in software production. The most likely explanation for their behavior is that U.S. computer producers have found that ISV entry results in a greater supply of software than would otherwise be available. The U.S. software industry has also benefitted from the prior existence of an enormous number of small contract programming companies that originated from the needs of larger companies and governments for custom software production. This infrastructure has played a major role in the growth of the U.S. packaged software industry, through its development of human capital (noted as a problem of the European industry in Chapter 7 of this volume) and its pioneering development of particular applications.[9] In the U.S. market, ISV participation appears to have fostered a faster sales growth than computer producers could have realized from joint production of hardware and software. For users, the presence of ISVs offers an alternative to internal production.

Change in the solutions to the "make or buy" problems of computer producers and users, solutions that are *unique to different stages of the of evolution of the U.S. computer market*, along with some key legal and policy decisions, explain the changing division of labor among producers of software in the United States. In other words, our working hypothesis is that there is no "natural" industrial structure for software production; the structure and evolution of a nation's software industry depends upon particular historical and institutional events. This chapter examines this hypothesis by exploring the development of the U.S. software industry, beginning with the period before its birth. During this period, the design and construction of computers as scientific instruments were co-mingled with writing computer instructions. I bring this discussion to the present day when independent software companies produce and sell packaged software for large installed bases of particular computer "platforms." Although it is tempting to simplify

---

[8] See Chapter 7 in this volume.

[9] For example, one of the pioneering relational data base products, dBase II, III, and IV, originated in a program developed at the Jet Propulsion Laboratory.

this account by discussing only the independent companies that produce software "products," that approach would obscure the enormous economic significance of software production by other organizations, including producers of hardware and the users of information technology. The specifics of history, institutions, and participants are important, and mean that the appropriate unit of analysis for a study of the software industry's evolution must begin at the national or regional level, rather than the "global market."

The outcome of these evolving make or buy decisions of computer users and producers in the U.S. market in recent years can be briefly summarized. Virtually all of the people involved in the information technology industry, from electronic component producers to users, write software. Software written for internal use within companies has historically been the largest single source of investment in software creation; this type (which is not separately captured in the U.S. national income accounts) is followed by software produced by design service and software firms, and software "embedded" in electronic systems.[10] In 1987, the receipts of U.S. software programming service companies (SIC 7371) were $14.2 billion, the receipts for computer integrated systems design (SIC 7373) were $7.1 billion, and the receipts from prepackaged software (SIC 7372) sales were $5.9 billion. The preponderance of revenues from programming services and a large share of integrated systems design revenues are derived from the production of "specialized" software solutions, that are unique to individual companies. By contrast, packaged software is an intermediate good or "tool" that providing applications solutions or entertainment for large numbers of users.

In assaying the size of the software industry, it is important to understand that the sale of software services and packaged software does not include the investment in software creating activities within organizations, the first of the three categories mentioned in the preceding paragraph. Some indication of the magnitude of these activities is available by analyzing the patterns of employment of software engineers, which are available for broad industry groups. For example, "business services" (SIC code 73) includes prepackaged software, software programming services, and computer integrated systems design, *as well as* other information processing-intensive sub-industries such as credit reporting, advertising, and mailing services. In 1990, the "business

[10] Unfortunately, embedded software cannot be covered in the present paper. Of the features discussed here, the only one of particular relevance to embedded software is the role of feedback in accelerating the advance of user interfaces, discussed in the section on the personal computer revolution below. Chapter 5 in this volume attempts to estimate the size of the "user-produced"software sector in Japan.

service" employment of system analysts and programmers was 74,000 and 130,940 respectively. In 1988 and 1989, by comparison, 217,310 system analysts and 264,110 programmers were employed in manufacturing and the other service industries (including state and local government). In other words, establishments outside the business service sector employed many more computer-related professionals (individuals who bear a major responsible for specifying and creating software) than did firms within the "software industry." Obviously, programmers and system analysts in SIC 73, manufacturing and all service sector activities (excluding SIC 73) are not engaged solely or even primarily in writing software. Nevertheless, the "in-house"employment of these occupational classes is so large that it raises the possibility that in-house software may account for a larger share of total U.S. software production than the output of firms producing for the (custom or packaged) market.

This chapter's sections treat the development of the software industry chronologically. Section two considers the origins of software creation and the early history of the software industry from 1950-1965, a period when computer producers dominated software production. The third section examines the critical 1965-1970 period when rapid growth in the utilization of computers, antitrust litigation, and delays in some key technological improvements fostered enormous growth in user-produced software and the birth of an independent software vendor sector. The fourth section examines two themes that shaped developments in the 1970s: growing dissatisfaction of users with the productivity and effectiveness of their internal software development efforts and the impact of smaller-scale "mini" computers, which altered both the distribution of software-creating activities and the use of computers within organizations. The fifth section discusses the role of the personal computer and the rise of "software publishing" and introduces the final element of the story to date, the role of consulting companies and information system integrators in addressing the problems of internal software development. The sixth and final section of this chapter summarizes these historical developments and identifies some of the forces that are shaping developments during the 1990s. This history addresses two elementary questions noted earlier--(1) What determines the division of labor in software production among hardware producers, computer users, and the companies that produce software as their primary business and the "independent software vendors" and (2) What key events in the history of the U.S. software industry have created a distinctly "American system" of software production?

The third and final question this chapter addresses is why has software persistently been identified as a "bottleneck" in the growth of information technology markets and as a drag on the

realization of productivity gains from utilization of information technology. This question has both a general answer dealt with briefly here in the introduction and specific answers for the U.S. that are main themes in the fourth and fifth section of this chapter. The basis for this question is the observation of rapid rates of growth in the level and share of software costs for information technology systems.[11] The growth of the cost share of software has been linked to the "craft production" techniques in the software industry that allegedly cannot match the pace of hardware performance improvement.[12] The purported result is rapidly rising costs of information technology due to the "bottleneck" of increasing software costs, a consequence may help explain the low measured productivity gains from investments in information technology.[13]

Do recent data on hardware and software expenditures support the "bottleneck" characterization? The sales of packaged computer software and services as a share of computer sales from 1970-1993 are depicted in Figure 1 using current dollar revenues.[14] During the

---

[11] See, for example, OECD [1985].

[12] See Baumol, Blackman, and Wolff [1989].

[13] See Roach [1987] and Strassmann [1990] for reviews of this issue.

[14] In addition to the information provided in Figure 1, other sources provide some evidence about earlier periods and suggest some qualifications to the empirical conclusions suggested by Figure 1. For information on the pre-1970 relative expenditures on hardware and software, see Phister [1976]. Phister reports results of a series of Datamation facility surveys that indicate that purchased software costs only became significant in 1967 when they were (0.1%) and gradually increased to 1.4% by 1974, the ending date of his report. More significantly, costs of system analysts and programmers accounted for 20.9% of system operating costs in 1955 and gradually rose to 32.3% by 1970 and remained near that level for the remaining four years of data he reports. Two snapshots are available in OECD [1985], pp. 82, 1979 and 1985. Between these two years, total software expenditures, more than doubled (from $51.9 billion to $113.3 billion (thousand million)) and the share of software expenditures increase from 15 to 19%.

There has, however, been some controversy over the extent of software cost increase. Gurbaxani and Mendelson [1987] note that the software costs reported to IDC (a leading consulting company) by data processing managers are a stable fraction of total EDP (electronic data processing) budgets. There are, however, three problems with this analysis. The first is that the growing deployment of computer hardware should engage economies of scale in software use for the shared systems under DP manager control, easing cost increases in this portion of the company's computer budget. Second, EDP budgets are often only a portion of the software investments since many companies have active departmental computing programs and the share of software in departmental budgets may differ from department data processing expenditures. Third, it is unclear how EDP department personnel costs should be assigned since many of these personnel are involved in programming activities. Since personnel costs are the most rapidly increasing portion of costs, accounting for internal software production would raise the share of software in corporate EDP costs for the years reported.

1970s, Figure 1 suggests that expenditures on packaged software and software services grew *at the same rate* as hardware expenditures. Thus, unless it can be established that the *amount* of purchased software utilized during the 1970s was *decreasing* (which is unlikely), price increases do not provide a plausible explanation for that decade's trends in software expenditure. In other words, for the bottleneck hypothesis to be valid, software use would have had to decrease. The microcomputer revolution of the 1980s accelerated growth in software sales, pushing software's share above 50% by 1988 and close to 75% by 1993. During the 1980s, the enormous growth of low-priced "packaged software" for personal computers mean that growth again cannot be explained with a cost-push or price-increase model that is consistent with the bottleneck view.[15] Outward shifts in demand and relatively elastic demand appear to perform better in explaining increasing expenditures on software than does a tightening bottleneck. This analysis, however, does not address the costs of internally produced software.

Although an inaccurate general characterization of U.S. industrial experience, the "bottleneck" problem has shaped the interactions among hardware producers, computer users, and independent software vendors, and has motivated some of the key institutional and policy events in the U.S. industry's development. Institutional reforms have been directed at sources of cost growth, particularly the development costs and maintenance expenses of *internally-produced* software. Thus, specific responses to the software bottleneck in the U.S. included changes in the organization of the delivery of information processing services such as the growth of system integration and outsourcing services as well as the deployment of specific types of hardware such as minicomputers. In both cases, efforts were directed at widening a bottleneck, although this bottleneck may be more accurately characterized as organizational rather than stemming from software *per se*.

[15] Even with very high costs in developing such programs as Lotus 1-2-3 or WordPerfect, the number of purchasers for these programs were unprecedented and increasing returns must surely apply to these products. Moreover, this is not an industry where the costs of failure can be passed on to consumers by increasing the price of some other product, instead they are borne by investors.

II. Early Developments: Origins until 1965

The development of computers during and immediately after World War II was directed toward scientific and technical rather than business objectives.[16] Like their electromechanical business machine precursors, early computers were programmed by rewiring and thus were highly specialized to particular information processing tasks.[17] After the war, Alan Turing and John von Neumann's ideas for stored program computer created the possibility of a general-purpose problem solving device that could be "programmed" to emulate and replace more specialized data processing machines.[18] Programmability and the possibility of "reusable"software gave general purpose computers an advantage over the large installed base of punch-card data processing equipment.[19] Maurice Wilkes, Director of Cambridge University's Mathematical Library expressed prescient views on the economic importance of reusable software: "There would be almost as much capital sunk in the library of sub-routines as the machine itself and builders of new machines in the future might wish to make use of the same order code as an existing machine in order that the subroutines could be taken over without modification."[20] In short, the problems of software "lock in"and the incentive for creating machines that could emulate the operation of earlier machines were present at the start of the industry.[21]

---

[16] Histories of the origins of the computer industry include Flamm [1988] on the government role in the creation of the computer, Bashe *et al* [1986] on IBM's early computers, and Williams [1985] on the technological history of computing devices (including precursors to the electronic computer).

[17] See Austrian [1982].

[18] The Univac I, Remington Rand's commercialization of a computer designed by Eckert and Mauchly, and the IBM 701 launched the first generation of general purpose commercial computers.

[19] In 1928, the five largest manufacturers of data processing equipment based on punched cards recorded profits of $32 million on sales of $180 million and the same companies recorded lower profits ($19 million) on an equal sales volume in 1937. The annual prewar revenues in this market were larger than the value of the installed base of general purpose computers in 1956, the era of the first generation (vacuum tube) computer. (Beniger [1986]).

[20] As quoted in Bashe *et al* [1986], p. 322.

[21] For an exploration of the role of lock-in as an influence on user choice see Greenstein [1991]. An explanation of how "lock in" was partially overcome through the growth of "niche" markets is developed in Bresnahan and Greenstein [1992].

An important early demonstration that computers could be used for purposes other than scientific computation was the development of the SAGE air defense system whose software requirements led to the founding of the System Development Corporation in 1956. The early commercial use of computers in the 1950s however, also gradually stimulated a market for software services. Producers of computer systems such as IBM provided programming services and software tools.[22] Providing these complements accelerated adoption of new general-purpose computers, reinforced links between computer producers and users, and laid a foundation for the reuse of software in future machines. If software instructions could be made less machine-specific, the costs of adopting new machines could be reduced.[23] Computer system manufacturers accordingly focussed on producing the tools for creating applications programs, rather than developing application programs themselves.[24] Corporations using computers thus needed to develop software for their own information processing applications. As noted above, computer producers have an incentive to stimulate the production of any and all software that will increase the value of computers and enhance their sales of computers. Accordingly, IBM also supported the formation of users groups such as SHARE, which, as the name implies, was devoted to the exchange of software routines.[25] Computer system producers that offered services and software to stimulate the use of computers, users that developed applications for their own use, and users that cooperated in the exchange of programming routines and methods formed the early economic organization for software development activities. The structure of this organization heavily favored the "make" rather than the "buy" choice in the acquisition of software.

Larger companies scaled up their in-house software development to utilize faster processing capabilities and substantial improvements in peripheral devices such as tape drives, printers, and

---

[22] For example, IBM supplied interpreters, programs that ease the problem of machine language programming by translating "assembly code" instructions into machine readable computer codes.

[23] Of course, this process also increases the substitutability among all types of computers. Programs such as operating systems (many of which *are* machine specific) therefore may play a special role in computer system manufacturers' strategies.

[24] IBM and its competitors provided significant assistance in software through the sales function. In IBM's approach, an application with significant value was identified and IBM and company engineers worked cooperatively to implement the software, often in cooperation with external contractors.

[25] Bashe *et al* [1986], pp. 347-349. While SHARE fell short of early expectations, it became important with the introduction of IBM's 704 in 1955 and accumulated a library of 300 programs.

disk drives. In-house development was facilitated by growing use of higher level languages. One of the first and most successful of these, FORTRAN (FORmula TRANslator), was introduced in 1957 and a 1958 survey of twenty-six SHARE user installations found that over 50% of these sites employed it for the majority of their problems, a rate of use that soon accelerated with further improvements in available compilers.[26] The substantial improvements in programmer productivity made possible by FORTRAN lessened the severity of bottlenecks in "in house" development efforts, extended the range of feasible applications, and freed users to consider new machines with compatible language features. Higher-level languages reduced the costs of "in house" development, further tilting the "make or buy" decisions toward the "make" outcome because they reduced the requirement for in-house creation of highly specialized (machine specific) "machine language" programs freeing programmer resources that would otherwise have slowed developments and forced many companies to employ external programming services.

While FORTRAN was used for wide range of applications, the demand for a high level language for accounting and other business applications was keenly felt by the user community.[27] These developments, as well as the sponsorship by the Defense Department of a committee to develop a "common business language," led to the specification of a new language COBOL (COmmon Business Oriented Language) in 1960[28] and two years later IBM offered COBOL for several of its 1401 models, including one of the smallest such systems.[29] Growth in COBOL usage was even more rapid than had been true of FORTRAN. IBM estimated that while FORTRAN use peaked in 1965, COBOL continued to grow rapidly growth through 1975, accounting for about 50% of software usage and displacing FORTRAN and use of assembler languages beginning in 1965.[30] The development of higher-level language support for IBM

---

[26] ibid., p. 357.

[27] ibid., p. 364.

[28] Flamm [1988]. See also Chapter 3 in this volume.

[29] Bashe et al [1986], p. 366.

[30] OECD [1985], p. 31. Assembly languages are machine-specific low-level languages that are closely related to the actual instructions executed by a computer, or "machine language." Higher level languages are "compiled" into machine language while assembly language programs are "assembled," a term that denotes the close translation between assembly and machine language. Although assembly language programs are typically more compact than the "compiled" version of a program can perform the same functions in less time, assembly language is also typically more time-consuming to write and debug than higher level languages such as FORTRAN or COBOL.

computers was an important factor in delaying the growth of an external market for computer software. Despite rapid growth in applications demand and centralized computer facilities, these higher-level languages supported productivity gains in software development that blunted the demand for external programming services of large organizations with in-house software developers.

In 1960, IBM introduced the IBM 1401, a less expensive general purpose machine addressing the needs of the medium size user. This machine was sold with a new high-level software language RPG, whose operations resembled those of punched card systems, and thus could be employed by individuals without costly retraining in the more abstract FORTRAN and COBOL languages.[31]

By 1965, IBM as well as its competitors including Burroughs and Control Data, had stimulated a market for programming services, software products, and professional services of $500 million in annual revenues. In 1965, the total value of shipments of U.S. computer manufacturers were estimated to be $2.4-$2.8 billion.[32] Much of the $500 million in revenues went to "service bureaus," companies that specialized in developing applications software such as payroll systems and selling information processing services to other, usually small, companies. By contracting externally for information processing services, client companies avoided investments in both computer hardware and software, albeit at the cost of having to redefine their information processing requirements to fit the solutions offered by service bureaus. One of the leading service bureau companies, Automatic Data Processing (ADP) was established in 1949 and by 1964 had revenues of $4.7 million, which grew to $20 million by 1968.[33] The McDonnell Automation Center, formed in 1960 and merged with McDonnell Douglas' California computer operations (following the 1970 merger of McDonnell and Douglas aircraft), recorded $47 million in revenue in 1970.[34] These, and other, computer service bureaus were competitive alternatives to the purchase of computer hardware and the in-house development or purchase of software. While many of these companies had strong sales growth in the latter half of the 1960s, the period was

---

[31] Bashe *et al* [1986], p. 480.

[32] Estimates of the value of domestic software and services market from CBEMA [1983] as excerpted in Juliussen and Juliussen [1990].

[33] Fisher, McKie, and Mancke [1983], p. 321.

[34] ibid., p. 320.

also marked by the rapid diffusion of smaller-scale computers such as IBM's 1401 that offered medium and smaller-sized organizations their own computer facilities while almost all larger business organizations had installed computers by the end of the decade.[35]

Early commercial applications of computers were associated with in-house programming using higher-level languages and the growth of service bureaus as an alternative supplier of computing services. This structure for the supply of software, in which computer manufacturers created "tools" for applications development, users developed application software, and a residual of users employed service bureaus for their data processing needs was short-lived. Developments that occurred from 1965-1970, including IBM's success with the System/360 *and* IBM's decision to unbundle software from its supply of computers,[36] increased the market for multi-installation software sales. The new entrants that formed the base of the independent software vendor (ISV) industry included software tool and utility program suppliers as well as "vertical market"software companies that provided applications for particular industries and for common software needs such as accounting systems. For these reasons, 1965-1970 were the pivotal years in the emergence of the current structure of the U.S. software industry.

[35] Phister [1976].

[36] IBM would also spin-off its service bureau operations to Control Data, DeLamarter [1986], pp. 87, 94.

III. Emergence--1965-1970

The number of independent software vendors grew rapidly during the late 1960s. Lawrence Welke, president of International Computer Programs, Inc. (ICP) testified in *U.S. vs. IBM* that, by 1965, 40 to 50 major independent suppliers of software and programming services and several hundred smaller organizations had been established.[37] Welke stated that these companies were deriving most of their revenues from work with the U.S. government and from systems development work on behalf of computer manufacturers.[38] The largest of these companies, Computer Sciences Corporation (CSC formed in 1959), had revenues of $5.7 million in 1964, $17.8 million in 1965, and $82 million in 1970. Much of CSC's growth in the late 1960s reflected the firm's development of multi-client software packages for accounting, ticketing, income tax preparation, commercial loans, and system operations. Similarly, Informatics, Inc., an early database producer began offering its product, Mark IV, in competition with IBM's unpriced "generalized retrieval system" in 1967.[39] By 1969, Informatics, Inc., revenues were $19.8 million from sales of computer service and software and there were over 170 installations with the Mark IV program.[40] Welke estimated that the 1965 population of 40-50 major and several hundred smaller companies had by 1969 grown to more than 2,800 such organizations. This enormous growth was focussed on contract programming services, which accounted for $600 million of these companies' revenues in 1969 which dwarfed their sales of $20-25 million in software products.[41]

Although many companies were founded prior to this period, 1965-1970 marks the beginning of the U.S. independent software industry. Before this period, software companies were small and relied on government contracts and system development work for hardware companies. In contrast with the service bureaus that were developing during the period, software companies had little direct contact with users other than the federal government and therefore had a difficult time marketing their services. Welke estimated that user expenditures on software creation skyrocketed from $200 million in 1960 to $3-4 billion in 1965 and continued upward to $8 billion

---

[37] Fisher, McKie, and Mancke [1983], p.322.

[38] ibid.

[39] ibid, p. 326.

[40] ibid.

[41] ibid., p. 323.

in 1970 and $12 billion in 1975.[42] Thus, with the contract programming and software sales of $625 million in 1969 noted above, users acquired less than 10% of their needs externally in the late 1960s.

Three events made it possible for independent software vendors to improve their position in the 1970s. The first was the 1964 introduction of the IBM System/360 "family." The System/360 unified the operating system software of the IBM product line, allowing users to develop software applications that could operate on systems ranging in monthly rental price from $8,800 to $60,000, with corresponding increases in computational power. The unification of software within the System/360 product line encouraged users to define a "migration path" from smaller to larger mainframe computers that would maintain the value of their software development effort as their use of software and computers intensified. These same opportunities supported the growth of computer service companies whose regional processing operations could be expanded through migration to System/360's with greater computational capacity as their business expanded. But the System/360 also provided the first instance of a broad "installed base" of computers with a single operating system. Independent software vendors had, for the first time in the industry's history, the opportunity to market the same product to a variety of users. By the late 1960s, the independent software industry still accounted for a small share of U.S. software creation activities despite rapid growth during the previous decade.

The second important event supporting the growth of the independent software sector was IBM's decision to unbundle the sale of hardware and software during 1969. IBM, which by various estimates accounted for between two thirds and three quarters of computer sales and installations, had previously offered software tools for creating user applications as part of the computer systems it leased and sold. Since IBM systems accounted for 66-75% of mainframe sales and leases during this period, its software "bundling" policy was highly influential. The practice was terminated on June 23, 1969 for new orders and January 1, 1970 for existing installations. The shock from this change was cushioned by IBM's announcement that it would continue to provide system software and previously produced applications and development tool software to both new and established users.

---

[42] ibid.

The motives for IBM's unbundling decision are disputed. One interpretation is that IBM's actions were made in response to anticipated litigation.[43] Fisher, McKie, and Mancke [1983] dispute this view, noting that no direct evidence of relation between the announcement and the DOJ antitrust action was discovered during subsequent litigation.[44] These analysts instead argue that IBM's costs of software support were increasing rapidly. The costs of developing the System/360 operating system software had proved a major trauma for the company, and seemed to foreshadow still further cost increases.[45] In addition, the growth of independent software vendors made it possible by the late 1960s for IBM to consider separate pricing for software and to retreat from its commitment to provide all of the software tools that users might need in order to purchase or lease IBM computers.

The third important event of the late 1960s for the software industry was the development of the minicomputer industry. Although IBM dominated sales of mainframe computers that occupied a central position in the data processing activities of their customers, centralization of computer operations had disadvantages. The inflexibility and inconvenience of mainframes imposed costs on users that wished to develop new applications based on real time computer control, research and development problem solving, and other specialized problems. For these users, the mainframe computer was a barrier to gaining exclusive and intensive access to computing time, achieving rapid turnaround in the development process, and linking devices such as scientific instruments under computer control. These problems persisted despite the frequent existence of excess computational capacity within mainframe computer installations.

Digital Equipment Corporation (DEC's) pioneered the minicomputer market with its introduction of the PDP-8 in 1965. The PDP-8 could be rented for $525/month, 6% of the cost of IBM's smallest System/360, the Model 30. The PDP-8 performed commercial type computational

---

[43] On December 6, 1968, IBM announced that significant changes in IBM marketing practices would be made in the following year. On December 11, 1968, Control Data Corporation (CDC) sued IBM for antitrust violations following two years of data gathering on IBM's practices that CDC shared with the Department of Justice. *U.S. vs. IBM* was filed a month later with claims similar to those offered by CDC. Brock [1975], p. 170. The protracted resolution of *U.S. vs. IBM* certainly prevented IBM from reconsidering its unbundling decision during the 1970s.

[44] Fisher, McKie, and Mancke [1983], pp. 176-77.

[45] See Pugh, Johnson, and Palmer [1991], pp. 331-345 who employ the term "trauma" to describe this experience. For a poignant summary of System/360 system software development see Brooks [1975], especially pp. 47-48.

tasks at about 6% and scientific computation at 22% the speed of the Model 30.[46] By compromising speed in order to achieve a very low cost, the PDP-8 tapped user needs that were not well served by the competing technological solution, time sharing, which had floundered in the late 1960s.[47] At one seventh the cost of a mainframe, users could afford to let the PDP-8 stand idle, dedicate it to a single task, or share it among a small team engaged in a development effort.[48] The minicomputer also extended the application of "real time" control, i.e. applications in which a computer directly controls scientific instruments or electro-mechanical systems.[49] Moreover, minicomputers could be used in dedicated data processing applications such as data collection and entry.

The ability to develop entirely new computer system architectures in which tasks were "distributed" provided a basis for networked computing, a means for combining computers of different sizes and computational capabilities and optimizing the system for both computational

---

[46] Both comparisons are drawn from Phister [1976], pt. 2, pp. 343-44 and pp. 350-351. The speed comparison is based on the machines' relative Knight index, a measure based on the speed of executing instructions in different types of applications where the frequency of particular instructions is used as a weight on their execution speed.

[47] Experiments with time sharing had begun when John McCarthy and his colleagues sought a means to address their own software development needs at MIT during 1960 and 1961, (Pugh, Johnson, and Palmer [1991], pp. 355-356.) IBM launched several experimental timesharing system software development efforts including the System/360 Time Sharing System (TSS) in 1967. Despite these efforts, the performance goals demanded of time sharing were not met. An history of IBM concludes, "Having been victimized by over-optimism, time sharing temporarily floundered during the last half of the 1960s." [Pugh, Johnson, and Palmer [1991], p. 364.] While part of IBM's problems with developing time sharing systems stemmed from failure to embrace computer architectures optimized to this application and the shortcomings of its early time sharing operating systems, companies such as General Electric that had overcome both hurdles fared little better. General Electric's 265 system was adopted by MIT and Bell Laboratories and was responsible for GE's leading position in the time sharing services market of the late 1960s. Nonetheless, this system (as well as IBM's) experienced accelerating decline in performance as users were added and, by several measures, was more expensive than batch computing on mainframe computers. (see Sharpe [1969], pp. 509-517.)

[48] Two contemporary viewpoints on decentralization were Tomaszewksi [1972], who advocated mobile "gypsy" development teams for centralized computers, and Wagner [1976], who argued that dedicated use of cheaper computers makes sense even with enormous unused capacity. Wagner's hardware-based solution prevailed, largely because of the reductions in hardware costs made possible by the minicomputer.

[49] Such "real time" systems had previously been developed for cost insensitive applications such as the control of military and space missions. The minicomputer allowed similar techniques to be employed in commercial applications.

performance and response time. First developed in the late 1960s, distributed computing had its greatest impact only during the 1970s. Minicomputers also made it possible for small organizations to begin to purchase and use their own computers. In a 1968 survey, U.S. manufacturing companies with fewer than 500 employees or less than $10 million in sales reported no use of larger computers.[50] In the same year about 30% of companies with 200-499 employees or $5-10 million in sales were using smaller computers, including the smaller System/360 mainframes and comparable machines including minicomputers. Applications and affordability were responsible for rapid growth in the minicomputer sector during the late 1960s, the period when time sharing operating systems for mainframes were floundering. By 1970, as the CBEMA estimates of Table 1 indicate, minicomputer unit shipments exceeded those of mainframes and, consistent with their price, were achieving about one seventh of mainframe revenues.[51]

To summarize the remarkable developments affecting the software industry of the late 1960s--IBM decided to unbundle hardware and software as its System/360 moved to a dominant position in the market for mainframe general purpose computers, independent software suppliers began to carve out a market by competing in quality with IBM software products, programming services and service bureaus were growing rapidly as they improved their self-developed software and benefitted from price per unit performance reductions in computer equipment, and the advent of the minicomputer created an enormous range of affordable commercial applications for computer systems. Although disentangling the relative contributions of each of these developments appears to be an impossible task, their collective impact, according to Welke, was to raise user software creation investments from the $3-4 billion level in 1965 to $8 billion in 1970. In the same five years, CBEMA estimates that software product and services revenue increased more than ten-fold from $200 million in 1965 to $2.5 billion in 1970, (see Table 2).

---

[50] Phister [1976], pt. 2, p. 453.

[51] Growth in U.S. shipments in minicomputers depends upon classification. Table 1 includes estimates of both Phister and the Computer and Business Equipment Manufacturers Association (CBEMA). Phister's inclusion of smaller and more specialized machines raises his estimate and pushes the date of first shipments of minicomputers back in time to 1957.

IV. Growth and its Discontents--The 1970s

By 1970, annual sales of computers and peripheral equipment (SIC 7573) were approaching $5 billion or nearly 4.75% of domestic business investment in equipment and structures.[52] Software product and service revenue were $2.5 billion in 1970, 50% of hardware revenues in that year. The entry of numerous software companies during the late 1960s had created an industry structure that was so fragmented that estimates of the number of firms vary from 1,500 to 2,800. Of the $2.5 billion in 1970 software product and service revenues, these firms may have divided less than $1 billion in revenue in a market that was still dominated by IBM and other system producers such as Burroughs and Control Data.[53] This implies that average firm size would have been $350,000 to $700,000, a revenue level that could support no more than about a dozen employees. In the early 1970s independent software development and sales was a "handicraft" industry" in which an extensive division of labor or specialization was impossible. Moreover, users faced enormous problems in sorting out the offerings and capabilities of these companies. This fragmentation of software and service suppliers helped sustain the growth of larger computer service companies such as Computer Sciences Corporation, McDonnell Douglass, ADP, and EDS.[54]

Early in the 1970s, improvements in hardware technology allowed faster "real time"access, an impetus for software industry growth. IBM introduced the System/370 in 1971 with new hard disk technology (the 3330 disk storage unit, also introduced in 1971) that made it possible for user online disk storage to exceed online tape storage for the first time in the history of the industry.[55] This development dramatically improved response time of time sharing systems, significantly increasing the performance of a system architecture that had been a great disappointment in the late

---

[52] To be precise it was 4.738%. The 1970 output of the computer and peripheral industry (SIC 7573) was $4,984 million while gross private domestic nonresidential investment $105.2 billion.

[53] Datamation estimated IBM's 1975 software revenues at 10% of its $11.1 billion revenues, Datamation [1976].

[54] During the 1970s, EDS began to provide on-site data processing services for large clients.

[55] Pugh, Johnson, and Palmer [1991], p. 532.

1960s and allowing the growth of time sharing service companies as a new source of software creation activities.[56]

As IBM's System/370 extended the capabilities of centralized data processing, Digital Equipment Corporation (DEC) and its competitors in minicomputers developed the market for decentralized data processing. Minicomputer and mainframe shipments had been approximately equal in number (but not in revenues) in 1970. By 1976, almost six minicomputers were sold for every mainframe and by 1980 more than ten minicomputers were sold for each mainframe (see Table 1). Each of these minicomputers required software, often of a very specialized nature requiring extensive user development efforts. For example, the oil refinery business was a major source of minicomputer use, sparking rapid expansion in the demand for complementary specialized software.[57]

Use of minicomputers as primary computers in smaller organizations, their use as "front ends" for mainframes, use of minicomputers in data communications systems, and their embedding in process control systems each required very different software. Although total demand for minicomputer software was very large, the diversity of minicomputer applications meant that the size of individual markets for "packaged" minicomputer software was limited. Small market size limited the economies of scale from software packages and led to the development of a minicomputer software industry structure that closely resembled that of the mainframe software industry, in contrast to the "mass market software publishing" that emerged with the personal computer in the 1980s.

The fragmentation of the minicomputer industry is indicated in Table 3 which reports on the results of *Datamation's* 1977 user survey of satisfaction with packaged software.[58] In the 1977 survey, 199 packages were rated by 5 or more users, the survey report's threshold of use for reporting the name of the software package. Mainframe package software applications dominated

---

[56] The enhanced computational capabilities of the System/370 were complemented by it use of integrated circuit memory, which proved to be faster and ultimately cheaper for storage of data and software instructions than ferrite core memory. (See Pugh [1984] for a detailed history of ferrite core memory, the critical technology for computer memory until the 1970s.)

[57] Petrochemicals was the leading sector in demand for minicomputers; installations grew from 300 in 1964 to 2,000 in 1974. See Phister [1976], pt. 1, p. 134.

[58] . In all, 5,813 responses reporting on specific packages were received from *Datamation* readers.

these "multi-user"packages, accounting for 72% of applications for which the computer system could be identified.[59] By comparison, non-IBM minicomputer package software, most of which was system software, accounted for only 10 packages in the survey, the same number of software packages that were listed for IBM's System/3 minicomputer. Collectively minicomputer software packages accounted for only 20% of the products mentioned by five or more respondents to the survey.[60] Thus, despite the far larger "installed base" of minicomputers, the relatively small number of packaged software products for minicomputers indicates continued fragmentation in the minicomputer software market as of 1977, despite rapid growth in the number of machines and users during the previous decade.

For the software industry as a whole, the 1970s were a period of broad-based growth accompanied by growing concerns about productivity in software development. Growth of software related activities was led by a ninefold expansion in data processing service revenues; but sales of software products and professional services also expanded by factors of 5.5 and 7 respectively during the decade (see Table 2). For mainframe computers, the 1970s were a period of intensive rather than extensive growth. By 1970, growth in the number of firms with mainframe computer systems, computers per dollar of revenue, and computers per employee had all decelerated from the high growth rates of the 1960s.[61] The organizations that had adopted mainframes by 1970 were developing application programs that would more intensively use their existing mainframe computers rather than increasing the number of mainframes used in their organizations. Growth in the use of minicomputers was both intensive (within firms) and extensive (in new user firms) during this period.

In both mainframe and minicomputer applications the problems of software development and maintenance received growing attention during the 1970s. Users were beginning to experience significant problems in managing their in-house programming efforts. The problems noted in the technical and management literature of the time include efforts by programmers to shelter their

---

[59] Of the 192 packages that could be linked to a particular type of computer system, 138 were mainframe software packages.

[60] Unfortunately, Gepner [1977] did not report user characteristics. He does report that 30,000 surveys were sent out for the 5,813 returns, a response percentage of almost 20%. Unless the mailing list was biased this is a large enough sample to capture a significant share of the minicomputer users of the period.

[61] Phister [1976], pt. 1., p. 129.

positions by creating software with high maintenance costs,[62] the intractability of deriving meaningful measures of software and system performance metrics for increasingly complex systems,[63] the hurdles of properly specifying large software systems,[64] the frustrations of organizational politics in managing such efforts,[65] and the growing disappointment with software quality.[66]

Problems with software did not prevent the 1970s from being a period of enormous growth in computers applications and perhaps contributed to employment growth in data processing. As early as 1972, computer equipment accounted for almost 23% of producer durables sold in the U.S.[67] Concerns about software noted above reflect continued growth in this investment, whose share of producer durables increased more than 33% by 1982.[68] Occupations related to data processing grew relative to both non-computer service and manufacturing sector employment

---

[62] For example, one software entrepreneur noted: "It is not at all uncommon for a programmer to threaten resignation, while simultaneously generating the type of undocumented programs that increases management's dependence on him. Thus he is in a position of strength from which he can (and, in the aggregate, often does) use mild blackmail to achieve greater status, money or dominance over management." (Brandon [1970]) See Kraft [1977] for the alternate view that programmer's work was becoming "deskilled"and routinized.

[63] See Kolence [1971] and Boehm [1973], [1981].

[64] Larsen [1973]

[65] Keen and Gerson [1977].

[66] The quality shortcomings of software creation activities were a frequent subject in the trade press and in conferences of the 1970s. For at the example at the 1974 National Computer Conference the President of the American Federation of Information Processing Societies "accused the data processing industry of ignoring the growing inability of the programming profession to produce enough good software." (See Dolotta *et al* [1976] for the quoted paraphrase of G. Glaser's remarks and the cite to Glaser [1974], the published version of the speech.) One example from *Datamation* in 1974 of many in the trade press, quotes Tom Steel of Equitable Life Assurance: "Quality is, of course, a complex attribute, but by whatever measure one chooses, current software scores low. It is usually inadequate functionally, inconsistent between actuality and documentation, error-ridden and inexcusably inefficient. Beyond all that, it costs far too much. I can think of no other products (aside, perhaps, from pornography and telephone service in New York) that have all these failing to anything like the degree found in software." (Kirkley [1974], p. 65.)

[67] CBEMA [1983], p. 14.

[68] *ibid.* It continued to grow until well into the 1980s, eventually leveling off at about one half.

throughout the 1970s.[69] New methods for organizing the growing investment in data processing equipment lagged behind the investment itself.

As the costs of supporting this new form of physical capital became more apparent, new structures for decision-making and oversight were needed. The symptoms of efforts to come to grip with these issues are the problems were increased attention to software and system performance, software specification, organization of software creation, and quality assurance. But because software was becoming the repository and mediator of information flows, productivity problems competed for managerial attention with the problems of extending access and utilizing a rapidly expanding flow of information.

[69] Baumol, Blackman, and Wolff [1989], Ch. 7.

V. The Revolution: Software in the 1980s

The 1980s were an extraordinarily complex period of evolution and growth for data processing. Before beginning a discussion of major new actors and activities, two major developments that distinguish the 1980s from the previous decade merit attention, the death of time sharing as it was known in the 1970s, and the retreat of computer manufacturers *except IBM* from software and service activities.

The process of creative destruction during this period razed an entire sector of the data processing industry during the decade, the time-shared service company. Table 4 and 5 report on the largest data processing companies in 1980 and 1991 that derived the majority of their revenues from software and services. In 1980, there were eight major (as well as dozens of smaller regional companies) timesharing services companies including Dunn and Bradstreet which is primarily identified with business services and software . By 1991, seven of these companies had fallen out of the Datamation 100, while the eighth (Dunn and Bradstreet) retained some online information services. Information service companies such as Prodigy (a joint venture among Sears, IBM, and other companies), Compuserve, and America Online have filled part of the vacuum left by the timesharing service companies, and the sale of processing time on remote computers is still an active but highly fragmented market. But remote computation services are now more closely tied to companies that offer system integration or vertical market (specialized by industry) consulting services. Thus, software creation activities for particular industry segments that might have been provided by timesharing service firms are now provided by more specialized service providers. These services may still be organized as a remote computing activity (e.g. ADP now receives much of its data from users electronically rather than on paper forms), but the 1970s-style time sharing company that offered a collection of applications within a single timesharing system has virtually disappeared.

During the 1980s the software activities of computer manufacturers also underwent dramatic change, as the importance of software and service activities declined significantly for computer manufacturers other than IBM. Table 6 shows software and services revenues for the major computer manufacturers for 1981 and 1991, excluding the IBM plug-compatible companies (e.g. Amdahl) that have never been prominent in software production. Computer manufacturers' total revenues from software almost doubled over the decade, but almost all of this increase was attributable to IBM; software's contribution to its total revenue grew from 17 to 20%. While DEC

expanded its software revenues as well, by 1991, software accounted only one-third of DEC's software and service revenue while IBM derived 84% of its 1991 software and services revenue from software.

U.S. computer manufacturers, with the exception of IBM, withdrew from software development during the 1980s through several different paths. For Unisys (formerly Sperry and Burroughs), NCR (now a division of AT&T), and Hewlett-Packard, 1991 software and service revenues fell below their 1981 levels and plummeted as a fraction of the firms' total computer-related revenues. Other firms, especially new entrants into workstations, minicomputers, and personal computers (such as Sun, Tandem, Prime, and Apple) specialized in selling hardware rather than software.[70] These developments reflected the maturation during the 1980s of the independent software industry. Sales of computers no longer required manufacturers to provide software other than a basic operating system. Instead, other companies (including the in-house development efforts of users) provided these complementary inputs.

The following sub-sections discuss three major developments of the 1980s that have affected the software industry. The emergence of the personal computer at the beginning of the decade provided a new and revolutionary organizing principle, mass publication of packaged software. The introduction of the workstation in the middle of the decade fueled the creation of a new, large class of software applications that exploited the workstation's sophisticated graphics and numerical computation capabilities. The productivity and organizational problems that first appeared in the 1970s supported the rapid growth during the 1980s of system integration companies and "outsourcing" of corporate data processing activities and management.

[70] An important qualification is that Apple Computer, Inc.'s operating system software for its Macintosh computers was bundled with the sale of the hardware and thus is not recorded as software revenue.

The Personal Computer Revolution

A succession of products introduced during 1975-81 was overshadowed by IBM's introduction of its PC in August, 1981.[71] IBM succeeded in learning from all of the experiments that had been undertaken in the first six years of the industry and introduced a machine that combined a reasonable level of computational power and an operating system that facilitated application development. Customers quickly endorsed the new product, purchasing over 13,000 of the machines in its first year.[72] While Apple and Tandy machines continued to outsell IBM for several years, IBM had attained 26% of the personal computer market by 1983. The market for personal computers grew very rapidly during the 1980s. Table 7 reports the IDC's data on U.S.-based computer industry shipments by value and volume for the decade. By 1984, personal computer sales accounted for more revenue than IDC's large, medium, or small scale computer market segments with shipments of 9.7 million personal computers for a total revenue of some $17 billion, bringing the installed base in 1984 to 23 million machines.

The rapid growth in the installed base of personal computers provided an homogenous market for operating systems and applications of unprecedented size. In 1984, the installed base of both large and medium size computers was less than 200,000 units and about 1.9 million small-scale systems were in use; there were 23 million personal computers in use in that year. The enormous size of the personal computer market created unprecedented scale and profit opportunities for software producers that were simply unavailable in the markets for larger computers, even though software in the latter markets often had higher price tags.

Three new software companies emerged in the early years of rapid growth in the personal computer market. As of 1985, Lotus with annual revenues of $226 million had become the 60th largest U.S. data processing company.[73] Microsoft at $163 million ranked 78th and Ashton-Tate at $110 million ranked 100th among U.S. data processing companies in 1985.[74] These companies were joined in 1988 by WordPerfect Corporation, whose sales in that year were $179 million.

---

[71] Langlois [1992] provides a concise history of the early development of personal computers prior to and including IBM's introduction of the PC.

[72] ibid., p. 23.

[73] Datamation [1986], p. 62.

[74] *ibid.*

Each of these companies revenues was dominated in 1986 by a single product that had penetrated a large share of the installed base of personal computers and dominated a class of software--spreadsheets for Lotus, operating systems for Microsoft, databases for Ashton-Tate, and word processing for WordPerfect.

While the arithmetic of corporate size in the personal computer industry is straightforward, it is less immediately obvious why a single product in any application class should garner a dominant share of personal computer installations. In practice, however, several network externalities can cause a single software product to become a *de facto* standard for an application.[75] First, there are advantages in establishing a single operating system standard. IBM's endorsement of the PC-DOS operating system provided Microsoft with an enormous advantage that it quickly converted into a position as the dominant supplier of operating systems for IBM and IBM-compatible personal computers.[76] Second, it is desirable for a user firm to choose a single application product within a particular class so that information created by one user may be shared by another. Lotus Development's spreadsheet application, Lotus 1-2-3, was chosen by many business users of IBM personal computers (for which it had been carefully optimized as Chapter 6 notes) and rapidly displaced the VisiCalc spreadsheet program. Third, the accumulation of skills, training materials, and add-on products that facilitate the use of the product may reinforce the dominant position of a software product. The development of numerous specific applications written for the dBase II and, later, dBase III and IV products of Ashton-Tate, as well as training materials and the accumulation of consultant skills in these products, are examples of such externalities. Fourth, widespread use of a specific software product creates externalities in the labor market; more individuals are available with skills in widely-used application programs. The widespread use of Lotus 1-2-3 in business schools and corporations made it possible to hire individuals who required no training in the use of the product, encouraging companies to expand their use of the product and erecting a substantial entry barrier to other products.

---

[75] See Chapter 6 for additional discussion. Network externalities increase the value of participation in a particular network. The telephone network is the archetypal example of network externalities; the value of a telephone connection increases with the number of others connected to the telephone network. A *de facto* standard is established by market outcomes as opposed to a *de jure* standard which is established by some deliberative process.

[76] IBM initially retained a proprietary advantage in the embedded software of the IBM PC that disappeared when small software companies succeeded in duplicating its functionality, a development that launched the PC-compatible or "clone" market.

Collectively, these effects have propelled specific applications to dominance in their class. Between 1981 and 1985, the share of the 15 largest independent software vendors increased from 37% to 72% of total personal computer software revenues. The three largest companies, Lotus, Microsoft, and AshtonTate, accounted for 35% of the total in 1985.[77] There are, of course, limitations to the impact of network externalities as the disappearance of Ashton-Tate through merger with Borland and the recent acquisition by of WordPerfect by Novell suggest. In particular, major differences in the functionality of products may fragment user choices and create fewer positive externalities.

Personal computer word processing software illustrates the limits of these externalities in permanently establishing market dominance. WordStar, produced by MicroPro International, originally competed against a number of products with more readily understandable user interfaces (such as MultiMate) or with more features (such as XyWrite). But WordStar and its competitors were displaced by WordPerfect, which provided extensive features and an attractive user interface. WordPerfect, in turn, has been unable to dislodge Microsoft Word as the primary word processing application on Apple's Macintosh and now faces major competition from the recent success of Microsoft's new operating system Windows, for which Microsoft Word is a commonly chosen word processing application. As I noted earlier, the acquisition of WordPerfect in March 1994 by Novell reflects the growing problems of competing against Word and Windows. Ashton-Tate's dBase products were challenged late in the decade by a new line of database products such as Paradox, leading to Ashton-Tate's acquisition by Borland, which had previously acquired one of Ashton-Tate's primary competitors. Thus, positive externalities reinforce the establishment of a single product standard for a class of personal computer applications, but they do not guarantee the emergence of a single standard.

The market in personal computer software resembled many aspects of publishing or mass entertainment. The similarity of promotion and distribution methods with book and record publishing was especially marked.[78] Use of independent distributors that could provide stocks of popular software to immediately satisfy demand for a "hit"product so that users needs could be

---

[77] Business Week [1986], p. 129.

[78] The Software Publishers' Association, established in the early 1980s, immediately began to publish information about the unit shipments of particular products and award "gold" and "platinum" status to bestselling products. (See Software Publishers Association [1988], [1989], and [1990] for recent examples.)

immediately satisfied became an important competitive weapon. The growth of personal computer magazines, with extensive advertising for software and hardware, further strengthened the similarities between this new software market and the recording or publishing industries. The retail distribution channels that had been established to sell personal computers also supplied their software while mail order suppliers or direct distribution provided a convenient source for more specialized products and served as discount outlets for the major products. Major software companies supplemented these promotion and distribution methods with direct sales efforts characteristic of earlier packaged software.

The economic advantages for a firm from creating and maintaining a leading software product were enormous. Most of the costs in the packaged software market were the fixed costs of product development. When amortized over millions of users, very large development efforts could be financed from current revenues. During the 1980s, the largest personal computer software companies invested 10-11% of their revenues in R&D.[79] Their development efforts led to ever more sophisticated products that could utilize increasing personal computer performance.[80]

High development costs have introduced elements of monopolistic competition into the personal computer software industry, as the costs of software development have begun reduce the high gross margins of producers.[81]

The huge size of the market for personal computers also created dynamic markets for complementary hardware products which in turn created new opportunities for applications markets. Markets for hard disks, display monitors, modems, and printers all grew very rapidly during the 1980s. The storage capacity of hard disks enabled software producers to deliver larger programs, and created a demand for utility software to maintain the larger collections of files stored on personal computer systems. Similarly, display monitors encouraged the creation of more

---

[79] Hodges and Melewski [1991].

[80] A major transition in the personal computer industry occurred as Compaq and IBM introduced new MS-DOS based computers based on the Intel iAPX 386 microprocessor. The iAPX386, also called the 80386, employed a memory addressing method that made it possible to develop software programs far larger than those that could be created for previous models of IBM and IBM-compatible computers. In combination with the greater availability of high capacity hard disk drives, the iAPX386 and its successors made it possible to devise much larger and more complex software products for the personal computer including Microsoft's Windows.

[81] Nonetheless, the gross margins of Lotus Development and Microsoft were 81 and 74% respectively as of 1989. These are impressive margins, even with the higher product development (R&D) costs of 14-15% experienced in the late 1980s. See Business Week [1990].

sophisticated graphic display programs and communication modems increased the demand for personal computer communications programs. Perhaps the most important development came from the improvement of printer technology. Canon's laser printing engine, used by Apple and Hewlett-Packard to produce laser printers, provided a raw capability to place images on a page at the resolution of 300 dots per inch. Many of the software programs of the late 1980s were devised to take advantage of this capability and an entirely new segment of personal computer, desktop publishing, emerged as a means to take advantage of the high resolution printing in the creation of announcements, newsletters, and other "print shop" quality documents, all of which could be produced by a single personal computer.

By the end of the 1980s, personal computer users were able to choose from thousands of programs for specialized applications and dozens of major software products for more general applications. The resulting software network is certainly as complex as the network of software applications that has been developed for all other types of computer systems. Although personal computers have been used as terminals for mainframe and minicomputer systems, limitations in transfer of information across systems have, until recently, limited the use of the personal computer as a node in the larger network of computational resources available in modern businesses. Recent efforts to "re-integrate" the personal computer into a corporate computer network that itself has become more "distributed" between mainframe and smaller computers is considered in the conclusion of this chapter.

The Workstation: Continuity and Change

The workstation, introduced by Apollo in 1981 and Sun Microsystems in 1982, was a hybrid between the personal computer and minicomputer. Like the personal computer, the workstation took advantage of dramatic reductions in the price per unit of performance of microprocessor integrated circuits. Like the minicomputer, the architecture and peripherals attached to the workstation's central processing unit provided high performance in computation and display. IBM compromised the computational capabilities of its PC to penetrate the mass market for desktop computers; by contrast workstation producers entering the market at roughly the same time sought to attract engineers and other technically sophisticated users who would otherwise use minicomputer or mainframe computers.[82] A major appeal of the workstation was its graphics capability. Graphics intensive applications involving computer-aided design (CAD) and computer-aided engineering (CAE) had been developed for minicomputers but suffered from limitations in graphic display capability and the fragmentation in the minicomputer software market.

The most successful of the workstation companies, Sun Microsystems, adopted a corporate strategy based on "open" standards involving the use of UNIX, a widely available operating system that had first been developed at Bell Laboratories. Sun's version of UNIX had been modified by U.C. Berkeley computer science researchers with support from the Defense Advanced Research Project Agency (DARPA).[83] UNIX was already widely used on minicomputers and Sun Microsystem's strategy was to persuade technical users and software developers that applications for its workstations would be "portable" to ever more powerful workstation products, imitating IBM's System/360 marketing strategy. Along with liberal licensing of its microprocessor architecture (see Chapter 4), this strategy proved enormously successful in inducing investment in software for Sun workstation products.

From 1985 to 1990 the number of suppliers listed in Sun Microsystem's Catalyst guide to software for Sun workstations grew from 177 to 1,325, despite a major revision in the Sun operating system that interrupted the growth of suppliers in 1989.[84] The majority of companies

---

[82] Langlois [1992] reports that IBM eschewed using the 8086 microprocessor for the IBM PC to avoid creating a machine that would compete more directly with other IBM products, p.22.

[83] See Flamm [1988] and Chapter 3 in this volume.

[84] From research notes by Carolyn Judy, Center for Economic Policy Research, Stanford University.

offered a single software product.[85] Their products were, in turn, often specialized, high-performance solutions to problems such as oil field management, molecular modeling, and electronic circuit design. The growth in applications software spurred further sales of Sun's workstations in the same virtuous cycle that supported sales of IBM PC and the Apple Macintosh personal computer products. A larger installed base stimulates software development, and the availability of software stimulates the purchase of additional hardware platforms compatible with that software. Although a similar cycle operated in mainframes and minicomputers, the relevant installed base of both workstations and personal computers was vastly larger, even in specialized markets, because of their lower price per unit of performance. The "virtuous cycle" dynamic thus operated with greater speed and economic impact.

[85] The percentage of vendors offering a single product were 68%, 67%, 69%, 66%, 61%, and 60% for each of the years 1985-1990 (Calculated from research notes provided by Carolyn Judy, see previous note). The lower fraction in 1989 and 1990 may reflect the change in the Unix used for Sun's operating system, a change that was supposed to encourage software development.

The Rise of System Integration and Outsourcing

The personal computer of the 1980s did not have the performance or capacity to replace mainframe computers and organizations continued to face the problems of maintaining and expanding mainframe software applications. Solutions to these problems appeared to require a sophisticated internal corporate business operation within the company whose major output was data processing services. Many companies discovered that these internal capabilities were an expensive drain on investment resources and, even more importantly, a distraction from the business activities necessary to the company's success. The continued growth in the intensity of computer operations during 1970s and 1980s provided a growing challenge for internal development capabilities. In many cases, internal bureaucracies ossified, or were simply unable to keep up with the pace of technological change. Accordingly, firms began to reconsider the make or buy decision for their entire data processing activity. If another firm could provide the technological knowledge and human resources to implement specialized software solutions, choose among complex competing hardware offerings, and deliver useful information services to internal users, why not buy these services rather than produce them in-house? The growing complexity of data processing technologies and markets pushed companies toward the "buy" solution and a number of companies emerged to satisfy this demand.

During the 1970s as was noted in Table 4, two companies had developed large system integration operations. These companies, Computer Sciences Corporation and EDS, originally established as computer service firms, delivered large scale system solutions to the federal, and to many state governments. By 1991, thirteen system integration companies had joined the ranks of Datamation 100, the largest U.S. data processing companies (see Table 5.) Among these companies were four of the big eight accounting firms, the 1970s leaders EDS and Computer Sciences Corporation, and seven other companies, most of which had been involved in providing computer programming and consulting services at a smaller scale during the 1970s. Among these companies, EDS is noted for the extensiveness of its intervention in building data processing departments, managing the requirements specification, subcontracting for software creation, and eventually staffing the day to day data center operations of client companies.[86] By comparison, most of the other companies have chosen a more limited role with regard to clients, alternately providing consulting, programming service, "change" management, or other specific services in varying proportions over time in partnership with client company data processing personnel.

[86] EDS also is active in more the market for more limited services.

The size and scope of these new outsourcing and computer service firms appears to signal an important change in the methods of developing large corporate information systems. Unlike the computer service bureaus or time-sharing services, software development and system design is performed in partnership or on behalf of single corporate users. This method of delivering programming services creates software that is not only organization-specific, but that also benefits from the supplier's multi-client business. The large computer service organization is able to finance the fixed costs of large scale software development and accumulation of specific technical competences in hardware and software that can be amortized over a large number of customers. The computer service organization can also have major impact on the supply of published software by negotiating multi-client site licenses and requesting specific software modifications for their clients' needs. Earlier computer service bureaus and timesharing services pursued economies of scale through investment in computer hardware and the delivery of "remote" computing; the new service organizations are achieving these economies by organizational economies of scale that allow "direct" delivery of their services. Their success, however, is likely to attract competition from computer manufacturers, software suppliers, and perhaps a new generation of timesharing services using improved data telecommunications.

Summing Up the 1980s

The 1980s were a complex period in the development of the U.S. software industry. The growth in mainframe and minicomputer applications and sales continued through the decade, but additional layers of complexity were introduced by the widespread adoption of workstations and personal computers. The variety and volume of hardware and software mushroomed, and so too did problems of compatibility and complexity in organizing and managing the much larger installed base.

The common theme of 1980s developments was the creation of methods for realizing economies of scale in the development of software. Personal computer software companies achieved economies of scale in software development with a "publishing"approach that tapped the immense installed base. The leading firms' positions were further reinforced by the positive externalities in skills and data compatibility. The large installed base of workstations and the use of UNIX as a common operating system supported development of specialized, computationally-intensive software. For mainframes, scale advantages were achieved in the creation of organization-specific software through the growth of the new service organizations. In each of these areas, software was developing characteristics of a mature industry with established actors, large scale organizations, effective distribution methods, and a stable population of users. Although user-produced software continued to absorb substantial resources, the viability and stability of the independent software vendor industry seemed assured.

The developments of the 1980s have many implications for the future of software creating activities, organizational design, and the future hardware and software markets. They also have played an important role in the U.S. software industry's international position. Beginning with a brief overview of this international position, the conclusion examines the implications of the growth of networking and the uncertain development of new techniques for software engineering, development, and maintenance.

VI. U.S. Software in the International Market and The 1990s: Signs of Reintegration? Network Growth and New Programming Methods

The rapid pace of change in hardware and software markets have given the U.S. software industry an advantage in international competition where. In packaged software, U.S. independent and system software producers holds very strong positions in domestic and foreign markets. According to IDC, one of the leading market research companies in the industry, the 1992 the U.S. domestic market share of U.S. independent software companies in packaged software was 58% and the share of U.S. system companies was 30%.[87] In Europe, U.S. independent software companies held a 60% share of this market, the sum of 34% from independent suppliers and 26% from system vendors. As Malerba and Torrisi note in Chapter 7 of this volume, software in Europe is often delivered by hardware manufacturers and service companies and therefore it is possible that U.S. companies hold a smaller position in the European market than the above figures would indicate. It is true, however, that European packaged software consumption is comparable to that of the U.S. and many of the U.S. service companies, such as Arthur Andersen, have strong positions in the European market. In Japan, U.S. independent software companies hold a 27% share of the packaged software market and system companies a 33% share. The overall level of packaged software sales in Japan is, however, only one quarter that of Europe and the U.S. for the reasons suggested by Baba, Takai, and Mizuta in Chapter 5. U.S. service companies have a smaller position in the Japanese market.[88]

The most obvious explanation for the international competitive position of U.S. companies is that they have enjoyed a first mover advantage in all of the software industry's market segments. European and Japanese computer production and rates of utilization have historically lagged behind those of the U.S. providing domestic producers with smaller markets and fewer economies of scale advantages than those available to U.S. firms. First mover advantages were generated not only be commercial activity, but also by government R&D policy and the early development of computer science education in U.S. universities. The importance of software for national defense systems led to generous U.S. government support for basic and applied research in software, often through the Defense Advanced Research Projects Agency (DARPA).[89] The Association for Computing

---

[87] IDC [1993].

[88] See Siwek and Furchgott-Roth [1993] for further discussion.

[89] See Chapter 3 for more discussion of U.S. government programs.

Machinery (ACM), a professional association, played an important role in developing curricular standards for college-level study of computer science and aided in the rapid growth of university courses and degree programs in software engineering.

By the end of the 1980s, U.S. companies were engaged in another major advance in information technology, the linking of personal computers into extensive networks. Electronic mail (e-mail), file transfer, and "work group" software applications provided one impetus for networking efforts. Another impetus was the continuing importance of mainframe computers as repositories of organizational databases "of record" that were updated, edited, and analyzed by users employing personal computers. In addition to inter-user and mainframe communications, networks provided companies with new opportunities for sharing software and computational resources and for servicing and maintaining software within the organization. U.S. software development in the 1990s will be influenced by these growing network capabilities, reinforcing the U.S. software industry's position in international markets where network developments are likely to follow a similar path in the future. One of the largest U.S. software companies currently is Novell, which provides the leading local area network operating system.[90] Many of the currently available database programs have introduced features that permit their use on networks (e.g. record locking which prevents two users from simultaneously attempting to change database information.) These developments are likely only the first attempts at deriving value from the rapid growth in networking.

Networks provide new entry points for the introduction of externally produced software a user organization. Instead of convincing each individual user to adopt a particular software solution, software companies can market applications for many users at a single site or within a single organization and thereby generate higher revenues from a single purchase decision. In many respects, this marketing approach resembles the direct sales approach of earlier periods in the software industry. But it also allows software producers to deal with the problems of coordinating software needs within a firm. As noted earlier, system integration and consultant companies currently are addressing these coordination and compatibility problems. To the extent that users find software-embodied applications that help them coordinate computer use in the network, the need for external services may decline. The result may be greater competition between system

[90] Another example of software developed specifically for networks include Lotus Development's Notes, which provides a means to distribute and comment on documents over networked computers.

integrator companies and larger software producers who can add consulting activities to their software development, assisting a company in decisions on the complementary purchases necessary to use their application or operating system software. This competition may influence the structure of the software industry as consulting companies and software producers merge or acquire one another.

Assuming that security problems do not block the further extension of networking in software and computer systems, the re-integration of personal computers with other data processing resources within organizations and the linkage of these resources through data communication pathways will become more important in the next decade. In principle, this process should result in new software systems linking manufacturing companies with their suppliers and customers to manage delivery schedules, improving the efficiency of ordering and pricing procedures, and reducing the costs of managing the flow of material and labor inputs. Such gains will require the development of new technical compatibility standards for software such as EDI (electronic data interchange) and the creation of software applications that facilitate inter-organizational links. It seems likely that these developments will ensure growth in the market for externally-developed software solutions at the expense of internal development efforts.

A second issue likely to grow in influence during coming years is alternative techniques for software development. The personal computer revolution has demonstrated that substantial advantages can be gained from the use of standard application programs. These applications programs, however, have not eliminated the need within user organizations to develop custom applications software tailored to specific organizational needs. How these applications are implemented, and whether they can benefit from new development techniques will be major concerns in coming years. For example, there are two current approaches for improving the custom applications software development process.[91] The first approach uses computer assisted software engineering (CASE) methods to simplify, document, and maintain software. The second approach uses a new class of object oriented programming languages (OOP) to provide standard modules for the creation of more complex software systems including application programs.

From an economic viewpoint, the two approaches suggest somewhat different principles. A main objective of CASE is reduction in the costs of creating custom software while maximizing

[91] See Cusumano [1991] and Nakahara [1993] for different views of the Japanese approach to the same problem.

the flexibility and heterogeneity of traditional approaches to this task. Expanded use of CASE will accelerate the growth of customized software applications by lowering development costs. But the flexibility that makes CASE attractive also may limit its application. Flexibility allows programmers to address virtually any problem but also reproduces some of the productivity problems of earlier custom software. For example, although CASE techniques simplify code generation, they do not necessarily encourage the use of well-tested code modules. Thus, CASE techniques can lead to errors and inconsistencies similar to those of custom programs where programmers frequently reinvent (often with error) the methods of performing commonly performed functions.

OOP presents an entirely different tradeoff. An "object" can be used across units within an organization and across organizations. Application programs are assemblies of objects, some of which are custom-coded for a particular organization's needs. A measure of flexibility is embedded within the design of each object. This approach can generate the same sorts of externalities that were created by the mass market for personal computer software, as frequently reused objects are optimized for specific environments and as users improve their skills in the use of such objects. Moreover, as the software network of the corporation integrates objects, the opportunity for external suppliers to enter is enhanced.[92] But realizing advantages and funding the costs of developing sophisticated objects requires widespread adoption of a limited number of object libraries. Independent software vendors will offer many different object libraries in an attempt to become dominant suppliers and unless a "shakeout" reduces their number, a classic monopolistic competitive equilibrium (where costs of differentiation absorb potential profits) is likely to occur. Such an outcome would divert resources from the incremental improvement of object libraries and user skills in their use toward efforts to secure user adoption of one of many competing object libraries. The latter goal is likely to result in object libraries that are more specialized to particular classes of users and the outcome will be similar to existing product differentiation within application programs. OOP may thus move the long-established tension between standardization and customization in applications software to a new level, without eliminating it.

[92] The intellectual property debates over "look and feel" and the alleged infringement by Microsoft Windows on Apple's operating system are only the beginning of these kinds of disputes as users seek to construct a web of interactions among their programs.

The contest between CASE and "object" approaches to software development is only one example of the new tradeoffs that flow from the enormous installed base of personal computers and the growing use of networks. The contest between efficient solutions derived from custom solutions and the broad adoption of "standard" approaches that rely on the economies of mass reproduction are of growing importance in explaining the rate and direction of technical change in software. A better economic understanding of the emergence of technical compatibility standards and the conflicts between variety-enhancing competition and cost-reducing coordination are needed to more fully understand these developments.

Future developments in the U.S. software industry are likely to be shaped by the mature character of the supply "infrastructure" for software creation and the large installed bases of such particular information technology systems as personal computers. The U.S. software industry emerged from a complex "infrastructure" involving the interaction of computer producers, users, and independent software vendors. These actors were joined, for a time, by the timesharing vendors and, more recently, by the computer service consulting companies. The complexity of this infrastructure provides the U.S. with an enormous variety of possible solutions to new challenges presented by changes in hardware and application needs. Although there is no fundamental reason why other nations cannot, in time, develop similar infrastructures for software creation, the U.S. has an enormous lead in this process that is likely to be maintained. Among the sources of this diversity were important events unique to the U.S. market, such as the early creation of an independent software vendor sector because of IBM’s software unbundling decision, the U.S. lead in creating new computer "platforms" along with rapid adoption of these platforms that have stimulated complementary software development, and the aggressive adoption of information technology by public and private U.S. institutions. These events have been reinforced by early U.S. development of computer science education and widespread investment in on the job training in the use of software systems, generous funding of software research and development by the U.S. government, and the enormous size of the maturing U.S. market for hardware and software.

We now know that large installed bases, and the expectation of large future installed bases, of computer systems enhances investment in software development. The existence of such software supports expansion in the sales of compatible computer hardware as Chapter 4 in this volume argues. Change in the software industry is likely to be driven by the proliferation of computational "platforms," such as widely accepted personal computer and workstation models, and the associated demand for large investments in software development to support application of

these platforms. Many "platforms" that will influence software development in coming years are not yet available. The recent introduction of personal digital assistants (PDAs), handheld computers that accelerate the trend toward portability, and plans for the development of customer equipment, including PDAs, that will allow access to services offered over a "national information infrastructure" of fibre optic telecommunication networks suggests that the coming decade may initiate another cycle of investment growth in software creation that involves all of the actors that have emerged to date and perhaps new ones as well.

References

Austrian [1982]
Geoffrey D. Austrian, *Herman Hollerith: Forgotten Giant of Information Processing*, New York: Columbia University Press.

Bashe et al [1986]
Charles J. Bashe, Lyle R. Johnson, John H. Palmer, and Emerson W. Pugh, *IBM's Early Computers*, Cambridge: MIT Press.

Baumol, Blackman, and Wolff [1989]
William J. Baumol, Sue Anne Batey Blackman, and Edward N. Wolff, *Productivity and American Leadership: The Long View*, Cambridge: MIT Press.

Beniger [1986]
James R. Beniger, *The Control Revolution: Technological and Economic Origins of the Information Society*, Cambridge: Harvard University Press.

Boehm [1973]
Barry W. Boehm, "Software and Its Impact: A Quantitative Assessment," *Datamation*, May, pp. 48-59.

Boehm [1981]
Barry W. Boehm, *Software Engineering Economics*, Englewood Cliffs, N.J.: Prentice Hall.

Brandon [1970]
Dick H. Brandon, "The Economics of Computer Programming" in George F. Weinwurm, *On the Management of Computer Programming*, New York: Petrocelli Books, p. 8.

Bresnahan and Greenstein [1992]
Timothy F. Bresnahan and Shane Greenstein, "Technological Competition and the Structure of the Computer Industry," Center for Economic Policy Research, CEPR Publication #315, Stanford University, June.

Brock [1975]
Gerald W. Brock, *The U.S. Computer Industry: A Study of Market Power*, Cambridge: Ballinger.

Brooks [1975]
Frederick P. Brooks Jr., *The Mythical Man-Month: Essays on Software Engineering*, Reading, Massachusetts: Addison-Wesley.

Business Week [1986]
Business Week, "Software: The Growing Gets Rough," *Business Week*, March 24, pp. 128-134.

Business Week [1990]
Business Week, "Software: It's a New Game," *Business Week*, June 4, pp. 102-106.

CBEMA [1983]
Computer and Business Equipment Manufacturers Association, *The Computer and Business Equipment Industry Marketing Data Book*, Washington, D.C.: CBEMA.

Cooper [1978]
John D. Cooper, "Corporate Level Software Management," *IEEE Transactions on Software Engineering*, Vol. SE-4, no. 4, July, pp. 319-26.

Cusumano [1991]
Michael A. Cusumano, *Japan's Software Factories: A Challenge to U.S. Management*, New York: Oxford University Press.

Datamation [1976]
Datamation, "The Top 50 Companies in the Data Processing Industry, *Datamation*, June, pp. 49ff.

Datamation [1981]
Datamation, "The Datamation 100," *Datamation*, June, pp. 91ff.

Datamation [1982]
Datamation, "The Datamation 100," *Datamation*, June, pp. 115ff.

Datamation [1986]
Datamation, "The Datamation 100," *Datamation*, June 15, pp. 43ff.

Datamation [1989]
Datamation, "The Datamation 100," *Datamation*, June 15, pp. 7ff.

Datamation [1992]
Datamation, "The Datamation 100," *Datamation*, June 15, pp. 12ff.

DeLamarter [1986]
Richard Thomas DeLamarter, *Big Blue: IBM's Use and Abuse of Power*, New York: Dodd, Mead and Company.

Dolotta *et al* [1976]
T. A. Dolotta, M. I. Bernstein, R. S. Dickson, Jr., N. A. France, B. A. Rosenblatt, D. M. Smith, and T. B. Steel, Jr., *Data Processing in 1980-1985: A Study of Potential Limitations to Progress*, New York: John Wiley.

Fisher, McKie, and Mancke [1983]
Franklin M. Fisher, James W. McKie, and Richard B. Mancke, *IBM and the U.S. Data Processing Industry: An Economic History*, New York: Praeger.

Flamm [1988]
Kenneth Flamm, *Creating the Computer: Government, Industry, and High Technology*, Washington, D.C.: Brookings Institution.

Gepner [1977]
Herbert L. Gepner, "User Ratings of Software Packages," *Datamation*, December, 1977, pp. 118ff.

Glaser [1974]
G. Glaser, "Keynote Address, 1974 National Computer Conference," Montvale, N.J.: American Federation of Information Processing Societies.

Gordon [1989]
Robert J. Gordon, "The Postwar Evolution of Computer Prices," in Dale W. Jorgenson and Ralph Landau, eds., *Technology and Capital Formation*, Cambridge: MIT Press, pp. 77-126.

Gordon [1990]
Robert J. Gordon, *The Measurement of Durable Goods Prices*, Chicago: University of Chicago Press.

Greenstein [1991]
Shane Greenstein, "Lock-in and the Costs of Switching Mainframe Computer Vendors: What Do Buyers See?," Faculty Working Paper 90-1718, Political Economy Series #48, University of Illinois.

Gurbaxani and Mendelson [1987]
Vijay Gurbaxani and Haim Mendelson, "Software and Hardware in Data Processing Budgets," *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 9, September, pp. 1010-1017.

Hodges and Melewski [1991]
Judith Hodges and Deborah Melewski, "Top 50: Profiles of the Leading Independent Software Companies," *Software Magazine*, June, pp. 23ff.

IDC [1992]
IDC, Computer Industry Reports: The Gray Report, Framingham, Massachusetts.

IDC [1993]
International Data Corporation, Application Development Tools: 1993 Worldwide Software Review and Forecast, Framingham, Massachusetts, December.

Juliussen and Juliussen [1990]
Karen Juliussen and Egil Juliussen, *The Computer Industry Almanac: 1991*, New York: Simon and Schuster (Brady Books).

Keen and Gerson [1977]
Peter G. W. Keen and Elihu M. Gerson, "The Politics of Software Systems Design," *Datamation*, November, pp. 80-84.

Kirkley [1974]
John L. Kirkley, "The Critical Issues: A 1974 Perspective," *Datamation*, January, pp. 64-67.

Kolence [1971]
Kenneth Kolence, "Software View of Measurement Tools," *Datamation*, January, pp. 32-38.

Kraft [1977]
Philip Kraft, *Programmers and Managers*, New York: Springer-Verlag.

Langlois [1992]
Richard N. Langlois, "External Economies and Economic Progress: The Case of the Microcomputer Industry," *Business History Review*, Vol. 66, Spring, pp. 1-50.

Larsen [1973]
Gerald H. Larsen, "Software: A Qualitative Assessment," *Datamation*, November, pp. 60-66.

Nakahara [1993]
Tetsushi Nakahara, "The Industrial Organization and Information Structure of the Software Industry: A U.S.-Japan Comparison," Center for Economic Policy Research, Stanford University, CEPR Publication #346, May.

OECD [1985]
Organisation for Economic Co-Operation and Development (OECD), *Software: An Emerging Industry*, Paris: OECD.

Phister [1976]
Montgomery Phister Jr., *Data Processing Technology and Economics*, Santa Monica: The Santa Monica Publishing Co.

Pugh [1984]
Emerson W. Pugh, *Memories that Shaped an Industry: Decisions Leading to IBM System/360*, Cambridge: MIT Press.

Pugh, Johnson, and Palmer [1991]
Emerson W. Pugh, Lyle R. Johnson, and John H. Palmer, *IBM's 360 and Early 370 Systems*, Cambridge: MIT Press.

Roach [1987]
Stephen S. Roach, "America's Technology Dilemma: A Profile of the Information Economy," New York: Morgan Stanley (Special Economic Study), April 22.

Sharpe [1969]
William F. Sharpe, *The Economics of Computers*, (A Rand Corporation Research Study), New York: Columbia University Press.

Siwek and Furchgott-Roth [1993]
Stephen W. Siwek and Harold W. Furchgott-Roth, International Trade in Computer Software, Westport, Connecticut: Quorum Books.

Software Publishers Association [1988]
Software Publishers Association, *Publishers' Software Sales Report: 1988 Annual Report*, Washington, D.C.: SPA.

Software Publishers Association [1989]
Software Publishers Association, *Publishers' Software Sales Report: 1989 Annual Report*, Washington, D.C.: SPA.

Software Publishers Association [1990]
Software Publishers Association, *Publishers' Software Sales Report: 1990 Annual Report*, Washington, D.C.: SPA.

Strassmann [1990]
Paul A. Strassmann, *The Business Value of Computers*, New Canaan, Connecticut: The Information Economics Press.

Tomaszewski [1972]
L. A. Tomaszewski, "Decentralized Development," *Datamation*, November, pp. 61-64.

Wagner [1976]
Frank V. Wagner, "Is Decentralization Inevitable?," *Datamation*, November, pp. 86-97.

Williams [1985]
Michael R. Williams, *A History of Computing Technology*, Englewood Cliffs, N.J.: Prentice-Hall.

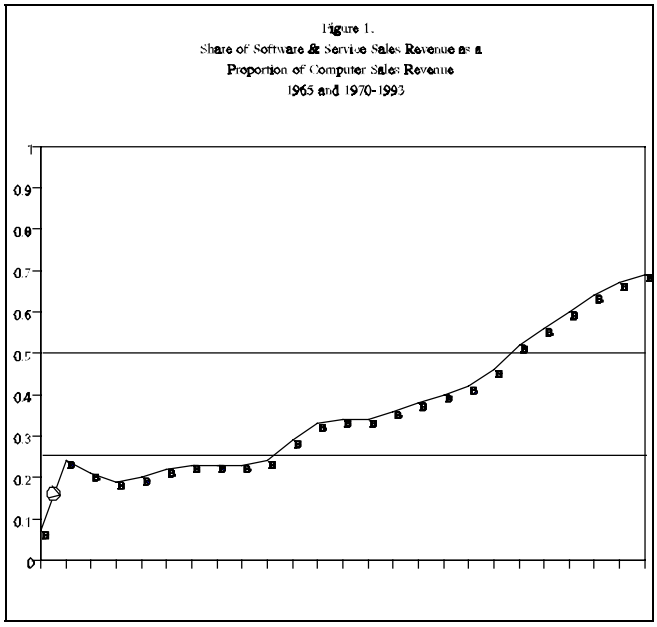
Figure 1.
Share of Software & Service Sales Revenue as a
Proportion of Computer Sales Revenue
1965 and 1970-1993
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

Table 1. Phister and CBEMA Estimates of U.S. Domestic Mainframe and Minicomputer Shipments by Number and Value 1960-1990

	Mainframe (nos.)	Mainframe ($ million)	Minicomputer (nos.)	Minicomputer ($ million)
Phister [1976] Estimates				
1960	1,500	560	300	30
1961	2,300	850	400	30
1962	3,100	1,060	400	30
1963	3,800	1,220	400	80
1964	5,100	1,570	500	100
1965	5,300	1,910	800	150
1966	7,000	3,200	1,000	130
1967	8,500	3,900	2,000	130
1968	7,400	4,650	3,500	185
1969	6,600	4,642	6,700	277
1970	5,040	4,073	9,500	282
1971	8,560	3,975	9,350	300
1972	10,970	5,170	15,100	450
1973	14,000	5,405	24,700	540
1974	8,900	6,220	34,000	810
CBEMA Estimates (from Juliussen and Juliussen [1990])				
1960	1,790	590	n/a	n/a
1965	5,350	1,770	600	66
1970	5,700	3,600	6,060	485
1975	6,700	4,960	26,990	1,484
1976	6,750	5,060	39,320	1,887
1977	8,900	6,940	56,780	2,780
1978	7,500	6,230	68,340	3,690
1979	7,200	6,340	81,250	4,712
1980	9,900	8,840	105,870	6,238
1981	10,700	9,640	121,990	7,290
1982	10,600	9,860	128,000	7,770
1983	9,980	9,780	146,800	8,979
1984	11,330	11,900	205,400	12,817
1985	10,910	11,890	190,800	11,696
1986	10,990	12,200	198,200	11,872
1987	11,200	12,660	205,800	12,080
1988	11,540	13,270	218,100	12,656
1989	11,890	13,790	227,700	13,093
1990	12,130	14,190	232,000	12,650

Note: n/a means not available

Table 2. CBEMA Estimates of U.S. Domestic Software and Service Revenues 1965-1988 ($million)

Year	Processing Services	Software Products	Professional Services	Total
1965	n/a	n/a	n/a	200
1970	1,200	500	800	2,500
1975	3,300	1,000	2,200	6,500
1980	10,800	2,850	4,350	18,000
1981	11,550	3,950	5,500	21,000
1982	12,650	4,900	5,950	23,500
1983	14,400	6,900	6,900	28,200
1984	17,150	10,000	8,100	35,250
1985	19,310	12,120	9,270	40,700
1986	20,750	14,150	10,100	45,000
1987	23,600	18,500	11,750	53,850
1988	26,900	27,850	13,300	68,050

Note: n/a means not available
Source: As cited in Juliussen and Juliussen [1990].

Table 3. Packaged Software and Computer Systems:
The Datamation 1977 Survey of Computer Software

Datamation, at that time the leading business publication in the data processing industry, mailed surveys to 30,000 computer installations and received 5,813 usable responses. Datamation then reported names and other information about software packages including user satisfaction for software packages mentioned by 5 or more respondents. Although 1,200 packages were mentioned by users, only 199 were mentioned by 5 or more users. The following tabulation is based on the information about computer system compatibility from the description offered. In some cases it was not possible to determine for which computer system the sofware package was designed and in many cases the class of computers is inferred from other information, e.g. operating system compatibility.

Computer System Type	Number	% of Total
Mainframe Computers	138	72
IBM System/3 and 32	20	10
Minicomputer Systems	19	10
Cross Platform or High Level Language	10	5
Other (e.g. IBM 1130)	5	3
Total for Percentages	*192*	*100*
Cannot Determine	7	
Total in Listing	*199*	

Source: Gepner [1977].

Table 4.
Independent Software and Service Companies--1980

Among the Datamation 100, the 100 largest U.S. data processing companies based on EDP revenue, 25 companies were *predominantly* engaged in the provision of software or services in 1980.

Type	Predominant Source of Revenue	Number
A	Software and Programming Services	4
B	Timesharing and On-Line Services	7
C	Systems Integration	2
D	Specialized Services	12

These 25 companies, by type, were:

Company	1980 DP Revenue ($ million)	Datamation 100 Rank	Comment or Specialized Service (if applicable)
Type A: Software and Programming Services			
SDC	187	43	Large software systems
Informatics	126	56	Large software systems
Dunn and Bradstreet	97	67	"Nomad" database product
Management Sciences America	53	96	Finance and Human Resource products
Type B: Timesharing and On-Line Services			
General Electric	475	17	
McDonnell-Douglass	280	25	
Tymshare	211	38	
Boeing	125	57	
United Telecom	115	61	
Comshare	88	68	
Martin Marietta Data Systems	78	74	
Type C: Systems Integration			
Computer Sciences Corp.	560	15	
EDS	408	18	
Type D: Specialized Services			
ADP	505	16	Accounting/Payroll Services
General Instruments	172	45	Wagering Systems
Bradford National	143	50	Financial Services
Planning Research	127	55	Government services
Reynolds and Reynolds	118	59	Auto dealer services
Shared Medical Services	106	64	Medical & hospital pharmacies
The Sun Company	87	69	Financial and disaster recovery
Interactive Data Corporation (subsidiary of Chase Manhattan)	69	81	Financial and data services
Commerce Clearing House	67	83	Tax preparation
Anacomp	57	89	Financial services
National Data Corporation	53	94	Cash management services
First Data Resources (subsidiary of American Express)	53	95	Credit card authorizations

Source: Datamation [1981]

Table 5.
Software and Service Companies--1991

Among the Datamation 100, the 100 largest U.S. data processing companies based on EDP revenue, 41 companies were *predominantly* engaged in the provision of software or services in 1991.

Type	Predominant Source of Revenue	Number
A	Software and Programming Services	17
B	Timesharing and On-Line Services	0
C	Systems Integration	13
D	Specialized Services	11

These 41 companies, by type, were:

Company	1991 DP Revenue ($ million)	Datamation 100 Rank	Category Specific Annotation
Type A: Software and Programming Services		Platform	
Microsoft	$2,276	12	Personal Computer
Computer Associates	1,438	23	Mostly Mainframe
Oracle	1,085	29	Mini and Mainframe
Lotus	829	40	Personal Computer
Novell	710	44	Network software
Dunn & Bradstreet Software	549	56	Mainframe
Wordperfect	533	57	Personal Computer
Borland	502	60	Personal Computer
Mentor Graphics	400	67	Specialized Workstation
ASK Computer Systems	395	68	Minicomputer
Cadence	353	73	Workstation
SAS Institute	295	79	Mini and mainframe
Autodesk	285	82	Personal Computer
Adobe	230	92	Personal Computer
Sterling Software	229	93	Mainframe
Legent	208	97	Mainframe
Information Builders	202	99	Mini and Mainframe
Type B: Timesharing and On-Line Services			
[no companies were listed for the 1991 Datamation 100. The new leading companies are Prodigy, Compuserve, and America Online.]			
Type C: Systems Integration			Parent Company
EDS	3,666	7	
Andersen Consulting	2,260	13	Accounting Firm
Computer Sciences Corp.	1,945	15	
Price Waterhouse	733	43	Accounting Firm
Science Applications Intl.	653	46	
Planning Research	623	49	
SHL Systemhouse	601	50	
Coopers & Lybrand	571	52	Accounting Firm
Martin Marietta	561	53	
Ernst & Young	551	54	Accounting Firm
Systematics Information	377	70	
MAI Systems	329	76	
Computer Task Group Inc.	285	81	

Table 5. (cont.)
Software and Service Companies--1991

Type D: Services			Predominant Service
ADP	1,933	18	General
American Express/First Data	995	33	Transactions processing
Nynex	601	51	Telecom and Financial
Bell Atlantic	550	55	Maintenance
Shared Medical	439	66	Hospitals
Policy Management Systems	342	75	Insurance
Boeing Information Services	325	77	Aerospace (Services to Boeing were 80% of company revenue)
American Management Systems	285	80	Telecom and Financial
Fiserve	281	84	Financial
Reynolds and Reynolds	233	90	Auto Dealers
National Data	221	95	Financial

Source: Datamation [1992].

Table 6. Software and Service Revenues
of Major Computer Manufacturers 1981 and 1991

Company (ranked by total DP revenue)	Software & Services Revenue ($ million)	Software & Services Revenue as % of Total Revenue
1981		
IBM	4,480	17.0
DEC	911	25.4
Control Data	1,154	37.2
NCR	1,029	33.5
Burroughs	838	28.6
Sperry	695	25.0
H-P	545	29.1
Honeywell	835	47.1
Xerox	209	19.0
Data General	130	17.0
Total	10,826	**22.**9

Company (ranked by total DP revenue)	Software+Services Revenue	Software+Services Revenue as % of Total Revenue	Software Revenue	Services Revenue
1991				
IBM	12,542	20.0	10,524	2,018
DEC	2,366	16.6	796	1,570
H-P	345	3.2	345	0
AT&T (incl. NCR)	850	10.4	250	600
Unisys (incl. Burroughs/Sperry)	1,200	15.0	600	600
Apple	250	3.8	250	0
Sun	175	5.1	175	0
Xerox	220	7.5	100	120
Tandem	140	7.2	110	30
Prime	247	17.9	247	0
Data General	210	17.3	210	0
Control Data	460	39.2	160	300
Total	19,005	**15.5**		

Honeywell’s computer division was spun--off and purchased by Bull of the U.K in 1987

Sources: Datamation (1982), Datamation (1992).

Table 7. U.S. Based Company Computer Industry Shipments by Value and Number 1980-1990

Year	Large-Scale Shipments by Number	Large-Scale Shipments by Value ($ millions)	Medium-Scale Shipments by Number	Medium-Scale Shipments by Value ($ millions)	Small-Scale Shipments by Number	Small-Scale Shipments by Value ($ millions)	Personal Computer Shipments by Number	Personal Computer Shipments by Value ($ millions)
1980	2,380	7,880	16,100	7,300	197,800	7,700	486,000	1,642
1981	1,510	6,150	19,200	9,160	212,500	8,800	905,000	2,707
1982	2,300	11,140	24,100	9,100	248,300	9,100	3,775,000	5,358
1983	3,390	14,460	28,000	9,530	282,390	9,800	7,623,000	11,304
1984	3,750	16,100	38,650	13,400	338,800	11,100	9,670,000	**17,168**
1985	3,240	16,970	39,650	14,610	325,400	12,320	8,828,000	19,070
1986	3,260	17,560	44,900	15,750	354,700	13,500	9,816,000	20,720
1987	3,380	18,340	49,900	17,500	391,200	14,980	11,130,000	22,650
1988	3,530	19,360	56,000	19,600	447,000	16,810	12,380,000	25,150
1989	3,780	20,520	63,000	21,850	512,300	18,720	13,500,000	27,800
1990	4,000	21,900	68,900	24,000	582,800	20,500	14,560,000	30,100

Source: IDC (1992)

By definition, software is a computer program that provides instructions and data to execute a userâ€™s commands. It is an indispensable part of the machine you cannot see, but it allows you to use the computer â€¦ just like how a mouse, monitor, hard drive and keyboard help you use the computer. Some common examples of software include Microsoft Word, Adobe Photoshop, Adobe Reader, Google Chrome, Gmail, Powerpoint, VLC, and many other similar computer programs that we often use in our daily life. If we sat down to list all the examples of software, the list would never end, but whatâ€™s more import All stories must be curated. Our content and digital storytelling services do this and more. We focus on framing and communicating your narrative in a meaningful way. This means analyzing your audience and using interpretive skills to find the most compelling, interesting, and persuasive information and the best medium in which to present it. Then, creative, thoughtful phrasing and design bring your story to life, resulting in a product that celebrates your accomplishments, inspires employees, engages the public, and frames your vision for the future.Â Our team can typically complete tasks like in-depth historical research, interpretive planning, and oral histories in less than six months. Small projects tend to take closer to a year, while larger projects may take two or three years. The software industry includes businesses for development, maintenance and publication of software that are using different business models, mainly either "license/maintenance based" (on-premises) or "Cloud based" (such as SaaS, PaaS, IaaS, MBaaS, MSaaS, DCaaS etc.). The industry also includes software services, such as training, documentation, consulting and data recovery . Drawing on interpretive policy analysis (IPA), the more. Although 98% of Turkeyâ€™s 3.6 million Syrian refugees live outside camps, municipalities lack formal authority to initiate policies, while receiving no government funding for refugees. Drawing on interpretive policy analysis (IPA), the article unpacks the empirical puzzle of how formally weak local governments respond to refugee needs. IPA expects policy to be constituted through diverse sets of local meanings. Case studies in three districts in Istanbul revealed distinctive local narratives, some of which consolidated the national age This brings us back to the external aspect of the SWOT Analysis method is the way that the External environment is perceived by a company and how it is analyzed. Being more specific, software companies have to pay really close attention to their external environment, since competition and market status play a huge role in the success of a software company.Â The U.S. Software Industry: An Analysis and Interpretive History. In D. C. Mowery, The International Computer Software Industry. 1995: Oxford University Press.