Development and Structure of the International Software Industry, 1950-1990

Martin Campbell-Kelly[1]
Department of Computer Science
University of Warwick

This paper is an attempt to develop a model of the software industry based on historical principles in order to complement a number of studies of the industry carried out during the last decade that were based solely on contemporary data and the very recent past. A study of the historical development of the software industry can shed light on several questions only partially addressed by the previous literature:

1) Why does the United States dominate the software industry? What are the historical reasons for the weak positions of the European and Japanese software industries? Does the developing nations' software industry represent a threat?
2) What role has R&D played in successful software firms?
3) Why did IBM and the existing software firms fail to penetrate the market for personal computer software in the 1980s?
4) Can recreational software and multimedia publishing be incorporated into a model of the software industry?

The software industry has existed since the mid-1950s, but until about 1970 very little attention was paid to it, largely because the

[1] I am most grateful to Robert Seidel, director of the Charles Babbage Institute, and Judy O'Neill for their illuminating comments on a draft version of this paper. My thanks also to Luanne James, former president of ADAPSO, and Walter Bauer, former president of Informatics, who provided information used in this paper.

BUSINESS AND ECONOMIC HISTORY, Volume 24, no. 2, Winter 1995.
Copyright ©1995 by the Business History Conference. ISSN 0849-6825.

industry was too small to merit detailed analysis other than as an unquantified sector of the overall computer business. As late as 1970, the annual turnover of all U.S. software firms was less than $.5 billion -- about 3.7 percent of the total computer business [Phister, 1979, p. 245].

The software industry began to grow significantly in the 1970s, first following IBM's 1969 unbundling decision and toward the end of the decade from the rise of the personal computer. By 1979 annual sales of U.S. software firms were about $2 billion. The 1980s saw dramatic growth rates in the software industry of 20 percent a year or more, so that the annual revenues of U.S. firms had grown to $10 billion by 1982, and to $25 billion by 1985 -- over ten times the 1979 figure.

As software gained in economic importance, the industry became the subject of several official inquiries in the mid-1980s, all of them motivated by anxiety over industrial competitiveness. The first inquiry, during 1983-84, was commissioned by the Information Computer Communications Policy (ICCP) Committee of the OECD and was published as *Software: An Emerging Industry* [1985]. The principal aim of this study was to move the software industry debate from the technical community to the arena of industrial policy in order to "identif[y] policy issues for governments" (p. 11). The main policy recommendations in the report concerned R&D investment, training, procurement policies, and standardization. Little attention was paid to the individual software industries of the member nations.

The OECD inquiry was followed by a number of domestic inquiries. The first of these was a lengthy and highly detailed report produced by the U.S. Department of Commerce, *A Competitive Assessment of the United States Software Industry* [1984]. This report acknowledged the supremacy of the United States in the international software industry and made a number of recommendations to ensure that the nation maintained its position. The recommendations primarily concerned improved intellectual property legislation, efforts to combat piracy, and the elimination of tariff barriers against U.S. software exports. No recommendations were made regarding the international competitiveness of the industry, except for a token suggestion about the need for tax incentives for R&D.

In Britain -- where the balance of trade in software had deteriorated during the 1980s -- the Advisory Council for Applied Research and Development (ACARD) conducted a domestic inquiry that was published as *Software: A Vital Key to U.K. Competitiveness* [1986]. This inquiry took a wider view than the U.S. study, addressing not only the supply but also the application of software. The ACARD study was very different in tone from the upbeat American report. The two main recommendations were first, that the government should use public procurement polices to foster a thriving U.K. software industry, and second, that there should be active government support for software engineering R&D and diffusion. These recommendations had a strong resonance with the government measures that had been taken to protect the British computer industry in the 1960s [Campbell-Kelly, 1989].

The ACARD report -- whose authors included a majority of scientists and technologists -- made little attempt at a structural analysis of the U.K. software industry, but instead focused on the need to improve software-engineering training and methodologies. Subsequently a much more focused analysis, *The U.K. Software Industry*, was undertaken by Peter C. Grindley of the London Business School [1988]. One of his main conclusions was that the industry's problems were not primarily technological; this point was forcibly and prominently made in the cover copy, which stated that "policies aimed to establish a major independent software industry in the U.K. founded primarily on software engineering ... are unlikely to be successful."

A comparison of these reports suggests two observations. First is the lack of historical perspective. For example, none of the reports attached much significance to a software firm's historical development in shaping its organizational capabilities and cultural perspectives, both of which intimately affect software artifacts. Moreover, although financial and market data were quite good for the 1980s, there was almost no attempt to collect data for earlier periods. Second, there was a marked inconsistency among the models of the structure of the industry. The OECD report, for example, divided the software industry into two sectors: the "hardware manufacturers" and the "computer services industry." The U.S. report placed firms into three categories: suppliers of "professional services," "software products," and "integrated systems." The ACARD report paid very little attention at all to industry structure, simply referring to "suppliers, appliers and users."

The Grindley report produced by far the clearest analysis of the industry, first by defining the concept of *tradable* and *non-tradable* software. This distinguished software sold as an artifact in its own right (such as a word processor or a database system) from software that was embedded in a product (such as the software component of a telephone exchange). The report then classified three major producers of software: "hardware manufacturers," "independent software producers," and "value-added retailers" (i.e., systems integrators). Even so, Grindley's concept of tradable software did not extend to recreational computer games; if it had, the Japanese firm Nintendo would (in 1992) have ranked above Microsoft in revenues [Feigenbaum, 1993].

In this paper I will present a classification of software firms based on their historical evolution. Three distinct sectors have evolved:

1) Software contractors
2) The packaged-software industry
3) The personal computer software industry

The software contractors were the first programming firms, the earliest dating from the mid-1950s. Their role was to develop one-of-a-kind programs for computer users and manufacturers. The ethos of the software contractors was analogous to that of civil engineering contractors in terms of their corporate culture, organizational capabilities, and relationships with customers.

The second group, the suppliers of packaged software, emerged in the 1960s to develop program products for computer users in public- and private-sector organizations. The packaged-software suppliers operated in direct competition with computer manufacturers, and like them, they have evolved the characteristics of firms in the capital-goods sector.

The third group, the personal computer software suppliers, became a significant sector in the late 1970s. None of the software industry reports of the 1980s emphasized personal computer software as a distinct sector, probably because its suppliers and the packaged-software suppliers appeared to be in the same business -- developing and selling multiple copies of programs. However, virtually all of the personal computer software firms developed outside the established software industry; in some cases there was a background in developing hobby or games software and in a "techie" computer culture.

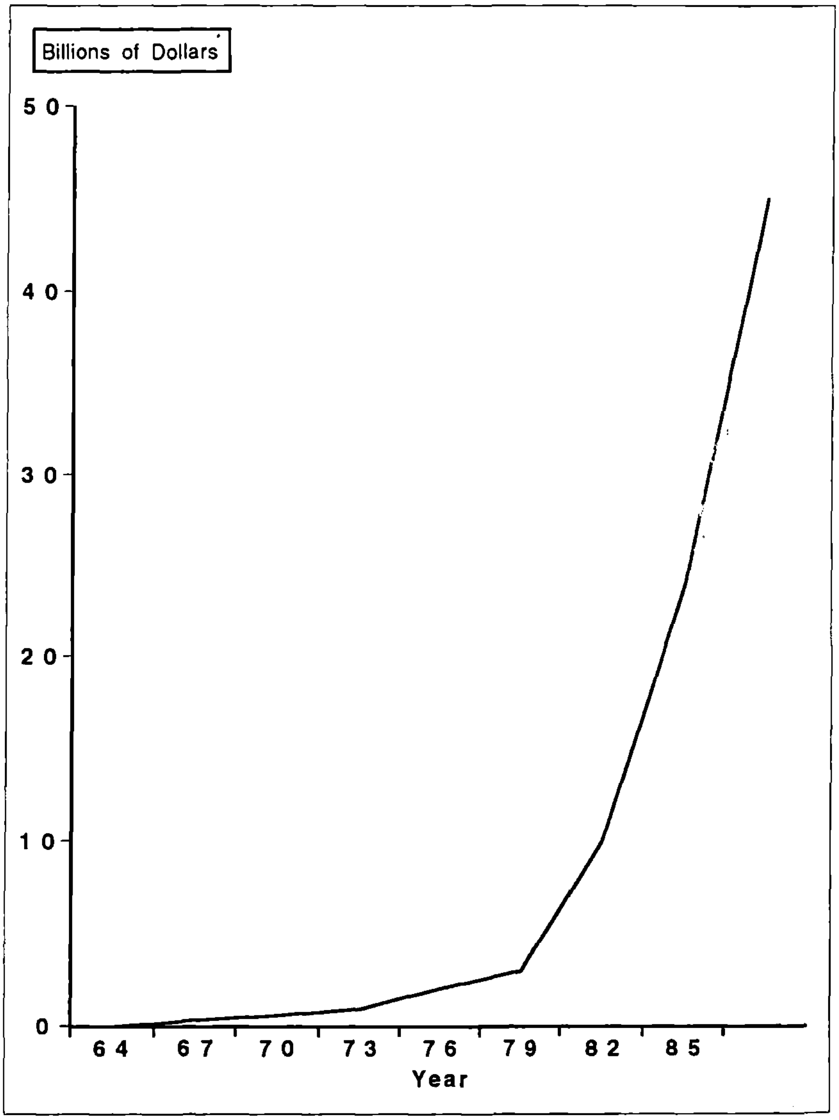
This background has profoundly affected the shape of the personal computer software industry and its products. Even though most personal computer software is now sold to corporate users, the ethos of the industry is more akin to publishing or consumer products than to capital goods. For example, for users of personal computers, software is an intensely personal issue -- and the relative merits of *WordPerfect* versus *Microsoft Word*, or *dBase IV* versus *FoxPro*, are debated with an almost religious fervor by their users. For the personal computer software manufacturer, the search for a "hit" product is paramount, leading to analogies with the popular music business or the Hollywood movie industry.

Personal computer software now dominates the public perception of software to such an extent that it is hard to keep in mind that it is only one of three roughly equal sectors in revenue terms. In particular, there is a tendency to leave IBM and the other computer manufacturers out of the picture and to put firms such as Microsoft and Lotus center stage. Yet, although Microsoft is the dominant supplier of software for personal computers, IBM remains by a factor of three the largest supplier of software overall. In 1992, IBM had a 31.9 percent share of the world market for software ($11.36 billion), compared with Microsoft's 8.3 percent market share ($2.96 billion). Microsoft was in fact the third largest supplier of software, narrowly beaten by Fujitsu's 9.9 percent market share ($3.52 billion) [Anon., 1993]. IBM operates in all three sectors of the software industry. It is a major software contractor, and it dominates the packaged-software market for IBM-compatible mainframes. IBM has the aim, so far unachieved, of competing with Microsoft as a supplier of operating software for personal computers. All the world's mainframe computer suppliers have a similar foothold in the software industry.

In attempting a history of the software industry, one is handicapped by the lack of data for the three decades 1950-1980. It seems plausible that this is one reason for the official reports' poor historical perspective on the industry. None of the reports has hard data prior to 1980; the best information is contained in a somewhat impressionistic graph "estimated ... from various sources including International Data Corporation" given in the U.S. report (Figure 1). Such was the paucity of data in Britain that the same graph was redrawn in the ACARD report (without attribution). The lack of statistical data on the software industry in the 1980s is reminiscent of that for the computer industry in the 1960s, when the OECD report *Gaps in Technology: Electronic Computers* [1969] deplored the lack of basic statistical data to make informed policy recommendations [p. 157].

In the absence of quantitative data, the best historical approach appears to be one based on case studies. Unfortunately, the existing literature on software is heavily biased toward internal, technical literature. For example, of the 150-odd items in William Aspray's "Annotated Bibliography of Secondary Sources on the History of Software" [1988], fewer than a handful are even tangentially related to the software industry. There appears to be only one full-length study of a software contractor -- Claud Baum's official history of the Software Development Corporation, *The System Builders* [1981]. The packaged-software industry is somewhat better served, with an excellent in-house history of Informatics [Forman, 1985]. Also worthy of note is Ben Voth's *A Piece of the Computer Pie* [1974],

Figure 1. Growth Curve of World Wide Software Revenues, 1964-1985



Note:
This curve should be treated as impressionistic evidence rather than as solid quantitative data. A revenue growth curve of this form first appeared in U.S. Dept. of Commerce [1984, p. 19]. It was subsequently reproduced in a similar form in Myers [1985] and ACARD [1986].

which describes a venture capitalist's participation in the University Computing Company. A more recent book, Hesh Kestin's account of Computer Associates, *Twenty First Century Management* [1992], is difficult to recommend on any grounds.

For the personal computer software industry, the literature is very different. One is faced with a glut of information about Microsoft on one hand and a complete absence of monographic studies on the rest of the industry on the other. Discounting the hagiographies of Microsoft's founder Bill Gates, only one book, Daniel Ichbiah and Susan Knepper's *The Making of Microsoft* [1991], stands out as a useful industry case study. Fortunately, the business press has taken an active interest in the personal computer software industry in the 1980s, and there are quite a number of articles on individual high-profile firms.

The Software Contractors

By the mid-1950s, the mainframe computer industry had taken on the form it was to keep for the next quarter-century: an oligopoly of U.S. companies, of which IBM was rapidly becoming the dominant player.

The term software had not yet been coined (it first came into use in about 1959), and computer programs were not a tradable commodity. Computer users obtained their programs in three ways. First, some computer manufacturers supplied (i.e., bundled) operating-system software as an integral part of the computer system. Because computers were so slow and small at that time (with typical speeds of 10,000 instructions per second and random access memories of around 10,000 characters), the system software could be developed at a modest cost with a small programming team. For example, IBM's FORTRAN -- one the largest system software projects of the mid-1950s -- was developed by a team of no more than ten people during 1954-57 [Rosen, 1967]. As well as system software, computer manufacturers provided programs for generic applications such as payroll, stock control, and report generation. The mainframe manufacturers also developed application program suites for specific industries such as retailing, manufacturing, banking, and insurance. This software was perceived by the manufacturer as a necessary part of the overall sales effort, and in some companies applications software was actually located in the marketing arm of the company.

A second source of software for computer users was the free exchange of programs within cooperative user groups such as SHARE for users of IBM computers and USE for Univac users [Armer, 1980]. These user groups, however, were populated mostly by technically oriented computer people who traded gossip and programming tips as much as actual software; there is no evidence of major competitive software systems being exchanged. IBM also facilitated the exchange of programs among computer users by maintaining a library of customer-developed programs.

The third source of programs was staff hired by a computer-using organization itself. The programming team would then develop in-house software when none was supplied by the manufacturer. This was an acceptable and often unavoidable expense in the context of the annual cost of a mainframe computer (typically $100,000 a year).

In the mid-1950s there were two gaps in this software supply situation that created two distinct markets for the newly emerging software contractors. One market -- which came primarily from government, large corporations, and the computer manufacturers themselves -- was for very large programs that the organizations did not have the technological capability to develop themselves. The second market was for "custom" programs of modest size for ordinary computer users who did not have the in-house capability or resources to develop software. The first market had significant technological and financial barriers to entry and was satisfied by large pre-existing firms. The second market, however, had much lower barriers to entry and offered an opportunity for the first software-contracting entrepreneurs. By the late 1960s, these two classes of entrants had become largely indistinguishable.

The first major entrant in the large-systems sector of software contracting was the System Development Corporation, which grew out of the RAND Corporation's participation in the SAGE air defense project. The SAGE system, developed between 1949 and 1962, was the first very large computer project, with a total cost eventually amounting to $8 billion. In 1952 IBM was awarded the contract to develop and manufacture the mainframe computers on which the system was to run [Bashe et al., 1986]. The SAGE software was estimated to need one million lines of code, which was an order of magnitude beyond IBM's or anyone else's experience. The contract for the SAGE software was therefore let to the RAND Corporation in 1955. Although lacking any actual large-scale software-writing capability, RAND was judged to have the best potential for developing it. To undertake the mammoth programming task, the corporation created the System Development Division, which became a separate entity, the System Development Corporation (SDC), in 1956 [Baum, 1981].

The SAGE software development program became one of the great heroic episodes in the development of software engineering [Bennington, 1956]. At a time when the entire stock of programmers in the United States was estimated at about 1,200 people, SDC employed a total staff of 2,100 including 700 programmers on the SAGE project. The central operating program amounted to nearly a quarter of a million instructions, and ancillary software took the total to over a million lines of code. It has been stated that the SAGE software development was a university for programmers. It was an effective, if unplanned, way to lay the foundations of the supremacy of the U.S. software industry.

In the second half of the 1950s and early 1960s several other major defense contractors and aerospace companies -- such as TRW, the MITRE Corporation, and Hughes Dynamics -- entered the large-scale software-contracting business. The only other organizations with the technical capability to develop large software systems were the computer manufacturers themselves. By developing large one-of-a-kind application programs for customers, IBM and the other mainframe manufacturers became (and remain) an important sector of the software-contracting industry.

IBM, for example, collaborated with American Airlines to develop the SABRE airline reservation system. The project began in 1954 and drew heavily on IBM's SAGE knowhow. The main software development took place during 1957-59 and was easily the largest civilian computer project to that date, involving two hundred technical personnel and producing a million lines of code. When fully operational in 1964, SABRE -- "the kids' SAGE" -- was said to have cost a total of $30 million [Burck, 1965; Gallagher, 1961; McKenney et al., 1995]. While still working on the SABRE project, IBM signed up Delta and Pan American in 1960; Eastern Airlines and several European companies, including BOAC, followed in the later 1960s. Most of the other mainframe computer manufacturers also became involved in software contracting, usually capitalizing on existing organizational capabilities inherited from their office-machine pasts: Univac specialized in real-time systems, NCR developed retailing applications, and Burroughs targeted the banks.

Although the mainframe computer manufacturers undertook major software contracts for their biggest customers, they did not have the resources or the small-scale economies to develop software for medium-sized customers. It was this market vacuum that the first programming entrepreneurs rushed to fill. Probably the first smaller software contractor was the Computer Usage Company, whose trajectory was typical of the sector [Kubie, 1994].

The Computer Usage Company (CUC) was founded in New York City in March 1955 by two technically qualified entrepreneurs who had acquired their knowhow with IBM. The financial barriers to entry were low because "all one needed was a coding pad and a pencil"; it was possible to avoid the capital cost of a computer, either by renting time from a service bureau or by using a client's own machine [Fisher et al., 1983, p. 322]. CUC was established with $40,000 capital, part private money and part secured loans. The money was used to pay the salaries of a secretary and four programmers who worked from the private apartment of one of the founders until cash flow was generated. The founders of CUC came from the scientific programming division of IBM, and most of their initial contracts were with the oil, nuclear power, and similar industries; in the late 1950s they secured a number of contracts with the National Aeronautics and Space Administration (NASA). By 1959 CUC had built up to a total staff of fifty-nine. The company went public the following year, netting $186,000, which was used to buy the firm's first computer.

By the early 1960s CUC had been joined by several other start-up software contractors including the Computer Sciences Corporation (CSC -- founded by a member of IBM's FORTRAN project team), the Planning Research Corporation (PRC), Informatics, and Applied Data Research (ADR). These firms were all entrepreneurial firms with growth patterns similar to the Computer Usage Company, although with varied application domains. Another major start-up was the University Computing Company in 1965 [Voth, 1974].

The first half of the 1960s was a boom period for software contractors. By this date the speed and size of computers, and the machine population, had all grown by at least an order of magnitude. This created a software-hungry environment in which computer manufacturers contracted out much of their own software development. CUC, for example, had a team of twenty programmers engaged on IBM's System/360 software development, and CSC was a major subcontractor to Honeywell for system software. Many private and government computer users were also contracting out big software projects. For example, in the public sector the defense agencies were sponsoring huge data-processing projects, while in the private sector banks were moving into real-time computing with the use of automatic teller machines (ATMs).

By 1965 it was estimated that there were forty to fifty major software contractors in the United States, of whom a handful employed more than a hundred programmers and had annual turnovers in the range of $10-100 million [Forman, 1985, p. 18; Fisher et al., 1983, p. 322]. CUC, for example, had grown to become a major firm, diversifying into training, computer services, facilities management, packaged software, and consultancy, in addition to its core business of contract programming. By 1967 it had twelve offices, seven hundred employees, and an annual turnover of $13 million. The specialist software contractors had also been joined by computer services and consulting firms such as Ross Perot's EDS, founded in 1962 [Levin, 1989], and Management Services America (MSA), formed in 1963 [Imlay and Hamilton, 1994]. There remains a considerable overlap between software contractors that have diversified into computer services and computer services firms that have moved into software contracting. This makes analysis of this sector of the industry always problematic.

Despite the rise of the major software contractors, the barriers to entry remained (and still remain) low. The major software contractors were the tip of an iceberg, beneath which lay a very large number of small software contractors, typically with just two or a few programmers. One estimate in 1967 stated that there were 2,800 software-contracting firms in the United States [Fisher et al., 1983, p. 323]. The only barriers to entry into software contracting for these small firms were a technical programming capability, knowledge of the targeted applications domain, and a client. However, there have been very few multinational entrants into software contracting since the 1960s -- which suggests that, though there are few barriers to small-scale software contracting, there are formidable barriers to embarking on major software contracts that involve hundreds of programmers. Only the existing firms have the necessary scale, access to the range of programming skills, applications knowledge, computer resources, and proprietary software development systems to embark on these projects.

Europe also developed some major software contractors during the 1950s and 1960s, generally a few years later than in the United States. In France, for example, the SEMA company was formed in 1958 as a joint venture between the Marcel Loichot management consultancy and the Banque de Paris [Lesourne and Armand, 1991]. The existing organizational capabilities of the firm lay in the area of industrial mathematics, operations research, and statistics. Early customers thus included the oil- and sugar-refining industries, the natural gas industry, the nuclear power industry, and the defense agencies. The specialized local knowledge required in many of these applications, or the fact that they were defense-related, effectively excluded any overseas competitor. Like its American counterparts, by the late 1960s SEMA had diversified into computer services and consultancy in addition to contract programming and had expanded into several European countries. In Britain, besides several computer services firms, two major software contractors were established in the first half of the 1960s, Logica and Computer Analysts and Programmers (CAP), both of which developed an international clientele.

The European mainframe manufacturers also developed significant contracting operations. In Britain, for example, ICL's major projects included the five hundred staff-year, £5 million LACES project for London Airport during 1967-71 [Harris, 1993]. ICL formed its own software-contracting subsidiary, Dataskil, in 1970.

The Packaged-Software Industry

The packaged-software industry took off in the late 1960s through a combination of economic, technological, and market opportunities.

The economic opportunity arose with the arrival of "third-generation" technology during the 1960s, when computer performance improved by two orders of magnitude (with processor speeds of up to a million instructions per second and memories of up to a quarter of a megabyte by the end of the decade); in the same period software productivity had improved by a factor of two or three at best. There was thus a growing gulf between the capabilities of computers to exploit software and its supply. Except for the very largest corporations, most computer users -- even if they had the technological capability -- now found it economically infeasible to exploit the potential of their computers by writing very large programs, because the development costs would be impossible to recover. This situation created an opportunity for existing software contractors to develop programs whose cost could be recovered through sales to ten or even a hundred customers.

The technological opportunity for the packaged-software industry was the simultaneous emergence of "the software crisis" and its panacea, "software engineering," during 1967-68. The essence of the software crisis was that the errors in and cost of writing software tended to grow geometrically with the size of a software artifact [Brooks, 1975]. The most notorious and well-documented example of the software crisis was that experienced by IBM in developing the system software for its third-generation System/360 computers, which was estimated to have cost a total of $.5 billion. The OS/360 operating system alone is said to have involved five thousand staff-years in its making, with a peak development staff of one thousand [Pugh et al., 1991, pp. 331-345]. For many users, writing big reliable programs was no longer a practical possibility: the choice was between paying a software contractor to develop a program and making use of a proprietary software package. The use of a package was very much more cost-effective than custom software. Sometimes a firm tailored the package to its business, but as often it adjusted its business operations to take advantage of an existing package.

Thus, particularly after the launch of the IBM System/360, which established the first stable industry-standard platform, a handful of software contractors began to explore the idea of converting existing software artifacts into packages. The economies of scope in writing custom software for different firms in the same industry, or in specializing in a generic application such as payroll, had already made that an established practice, but packaging took the process a stage further and was reflected in the price. In 1967 the market for software packages was still in its infancy, however, with only forty-seven products in the *Software Newsletter* of International Computer Programs (ICP - a software listing agency). And as late as 1970 packaged software sales were just $70 million, compared with $650 million for software contracting [Bauer, 1971, p. 55].

The development of the packaged-software industry was dramatically accelerated by IBM's December 1968 decision to "unbundle" its hardware and software--that is, to price them separately. It has never been established whether ordinary commercial judgment or antitrust pressure lay behind the unbundling decision. According to IBM sources, the decision was made because of rising software development costs. Between 1960 and 1969, the software component of IBM's R&D costs rose from about one-twentieth to about one-third of the total [Flamm, 1988, p. 24]; IBM's customers paid for this software whether they used it or not, and the step is said to have been taken to release customers from this "onerous" obligation. According to Fisher, McKie, and Mancke, who were consultants to IBM's antitrust team:

> IBM's announcement, on December 6, 1968, of its intention to unbundle came before the filing of the government complaint in early 1969, but after the Antitrust Division's investigation had been underway for some time, as had the consideration of unbundling. No evidence in the trial records suggests that the two events were related [Fisher et al., 1983, p. 177].

This opinion has to be set against the observation that a number of mainframe companies sought competitive advantage over IBM by remaining bundled for a few years while price structures re-established themselves. But by the mid-1970s, unbundling was the norm.

It took about three years for the packaged-software market to become fully established after unbundling. No doubt a thriving packaged-software industry would have developed eventually in any case, but the unbundling decision accelerated the process by transforming almost overnight the common perception of software from a free good to a tradable commodity. For example, Yates [1995] has noted that, whereas in the 1960s the American insurance industry had largely made do with IBM's "free" software, unbunding was “the major event triggering an explosion of software firms and software packages for the life insurance industry”; by 1972 there were eighty-one vendors offering 275 packages for the life insurance industry alone. Several computer-services firms and major computer users also recognized the opportunity to recover their sunk development costs by spinning off packaged-software products. The OECD report [1985, p. 52] observed that “noteworthy instances are Boeing, Lockheed and McDonnell Douglas in the United States, Imperial Metal Industries, Pechiney Ugine Kuhlmann, Elf-Aquitaine and Alitalia in Europe.” In the United States, software firms began to organize themselves as the Association of Independent Software Companies (ASIC); this short-lived organization was taken over by the Association of Data Processing Service Organizations (ADAPSO) in 1972, which then represented both computer-services and software firms.

Perhaps the most spectacular beneficiary of the new environment for packaged software was Informatics, the developer of the top-selling *Mark IV* file management system [Forman, 1985]. Informatics was founded in 1962 as a software contractor by former TRW personnel. In 1964 the company took over the software-contracting operation of Hughes Dynamics and in so doing acquired a file management system (an early form of database) known as Mark III. At this time the database offerings of the mainframe manufacturers were very weak, so Informatics decided to turn Mark III into a marketable product, *Mark IV*. They estimated the development costs at $500,000. Venture-capital firms had not yet become interested in software, so Informatics secured sponsorship from a heterogeneous group of firms (the Prudential, Standard Oil, National Dairy Products, Allen Bradley, and Getty Oil), each of which put up $100,000. *Mark IV* was launched as a product in 1967; there were few precedents for software pricing at this time, and its purchase price of $30,000 “astounded” computer users long used to obtaining software for free. By late 1968 it was only a modest success with 44 sales. After unbundling, however, growth was explosive -- 170 installations by spring 1969, 300 by 1970, and 600 by 1973. *Mark IV* now took on a life of its own, and even had its own user group -- called, predictably enough, the IV (Eye-Vee) League [Bauer, 1995]. Also in the wake of unbundling, Informatics was able to make the development costs of *Mark IV* an inventoried software asset that appeared on the balance sheet. *Mark IV* was the world's most successful software product for fifteen years until 1983, by which time it had cumulative sales of over $100 million.

Other firms, however, failed to make a smooth transition into packaged software. The Computer Usage Company, for example, effectively abandoned its $10 million a year software contracting operation, not in favor of packages, but to develop "turnkey" products -- selling a combination of hardware and a proprietary software package to a particular industry sector. Turnkey systems offered a middle road between pure software packages and software contracting. However, supplying hardware alongside software required organizational capabilities that the firm failed to develop. CUC had heavy losses in the second half of the 1970s, eventually going bankrupt in 1986. The University Computing Company also stumbled in the new marketplace; by diversifying into networks, it too found itself in the hardware business. By 1974 it was suffering heavy losses, from which it took a decade to recover. SDC also made a difficult transition. It took a number of its existing applications and tried to turn them into packages -- the first of them its *Text II* typesetting package for the newspaper industry. That industry was never sufficiently uniform to create a standard product, however; about fifteen systems were sold, but most of them required extensive tailoring that had not been built into the cost. Eventually SDC returned to its core business of software contracting, but it took most of the 1970s to recover financially; in 1979 it was acquired by the Burroughs Corporation.

There were relatively few entrepreneurial start-ups in packaged software, because in order to enter the market one needed to have a fully developed product, and this could be very expensive. The most active market was in database systems (in which IBM's offerings were weak), for which development costs were estimated at as much as $10 million [OECD, 1985, p. 43]. In general, the new packaged-software firms were organized by technically sophisticated entrepreneurs with access to venture capital. One of the first and most successful was the Cullinane Company, founded in 1968 by John Cullinane, a former IBM database expert. The Cullinane Company was typical of the new packaged-software entrants in that it was totally product-oriented and did not diversify into software contracting or computer services. Other new entrants into the database market in the mid- to late 1970s included Cincom, Software AG, and Oracle. The database-system suppliers of the 1970s conform very closely to the capital goods-supplier model of the packaged-software industry. Database software was used exclusively in the corporate context, its customer base was modest (typically a few hundred installations), and the product was technically sophisticated. Advertising for database products was conducted exclusively in trade magazines and through direct mail.

By the mid-1970s, all of the existing computer manufacturers were important members of the packaged-software industry. IBM was, of course, a major player from the day it unbundled. Despite its often mediocre products, it had two key advantages -- inertia sales to the huge base of pre-existing users of its software, and the ability to lease its software for a few hundred dollars per month. Leasing gave IBM a great advantage over the rest of the software industry, which did not have access to leasing funds and needed outright sales to generate cash flow.

Of the independent packaged-software producers, the most successful was unquestionably Computer Associates, which was formed in 1976 by former executives of the University Computing Company. Computer Associates initially occupied a niche supplying a sorting program for IBM mainframes. Computer Associates was one of the first major computer software firms to make a corporate strategy of growth through merger and acquisition. All Computer Associates' moves were aimed at acquiring "legitimate products with strong sales" rather than technological capabilities -- indeed Computer Associates typically fired half of the staff of the companies it acquired [Kestin, 1992, p. 15]. By 1987 Computer Associates had taken over fifteen companies, including the University Computing Company (then called Uccel), the world's second largest software company, for $629 million. Numerous other acquisitions followed, including ADR, Panasophic, and Cullinet. By 1992 Computer Associates had annual revenues of $1.4 billion, placing it fifth among software companies worldwide. Computer Associates was the only member of the old-line packaged-software suppliers to make the transition to the new market of personal computer software, and it did so by acquisition.

The Personal Computer Software Industry

The personal computer software industry effectively began in 1975-78. Apart from its meteoric growth, the most remarkable aspect of the industry has been its almost total disconnection from the previously existing packaged-software industry -- which in 1975 was a $1 billion-a-year business with several major international companies.

A distinct personal computer software sector developed as a result of the low financial and technical barriers to entry and the failure of the existing software firms to recognize the potential for a personal computer software market. That failure in turn stemmed from the origins of personal computing in 1975, as a pastime for young, male technophiles, with a culture akin to that of amateur radio enthusiasts. In 1975, the first personal computer (the Altair 8000) was advertised to hobbyists in the popular technical press and sold by mail order for construction from parts [Frieberger and Swaine, 1984]. The computer was programmed by hand switches and did not do anything remotely useful.

During the period 1975-78, however, the personal computer evolved very rapidly into its classical configuration of a central processing unit equipped with a keyboard for input and a visual display unit for output (at first often a television receiver). During this period several manufacturers, including Atari, Apple, Commodore, and Tandy, began to make and distribute machines in quantity. At this time the line between personal computers and video games was blurred, with Atari, Commodore, and Tandy exploiting that sector of the market at least as actively as that for personal computing. The personal computer market thus came to be associated with consumer electronics, and it was natural that the existing corporate packaged-software suppliers would not perceive this market as an opportunity.

The financial and technical barriers to entry into the personal computer software industry were initially low because the specifications of an early personal computer were low. The personal computer of 1975-78 had roughly the capability of a mainframe of the mid-1950s (for example, having a memory of 16-64 Kbytes). The key organizational capabilities of the existing software firms, with their powerful tools and methodologies for developing large, reliable software systems, were irrelevant for developing programs for the tiny memories of the first personal computers; indeed they were likely to be counterproductive. Entrants to the new industry needed not advanced software-engineering knowledge, but the same kind of expertise as the first software contractors in the 1950s -- creative flair and the technical knowledge of a bright, first-year computer science student.

It is important to note that, although the existing software companies were oblivious to the personal computer, they were well aware of the potential of the microprocessor -- the chip at the machine's heart. They saw the microprocessor, however, as a component in an integrated system (for example, in photocopiers, faxes, and other smart machines). Software for such "embedded systems" tended to be developed with sophisticated software tools on full-size computers. The technological and cultural gulf between the two views of the microprocessor was enormous. Policymakers were no better than participants in the existing software industry at recognizing the opportunity represented by the personal computer. Thus both the OECD [1985] and ACARD [1986] reports devoted far more attention to embedded systems than to the burgeoning personal computer industry.

The PC software industry can be seen to have developed in two phases. The first phase, which can be characterized as the gold-rush era, lasted from about 1975 to 1981; during this period, barriers to entry were extremely low, there were several thousand new entrants, and firms developed through organic growth. The second phase, which began about 1982 following the standardization of the PC market around the IBM-compatible platform, was a period of consolidation in which many of the early firms were shaken out, new entrants required heavy inputs of venture capital, and a small number of (American) firms emerged as global players.

The most prominent of the first wave of PC software firms was Software Arts, which produced the *VisiCalc* spreadsheet. *VisiCalc* was developed through part-time work by two young Harvard MBA students, Daniel Brinklin and Bob Frankson, for the Apple II in 1979; it was reported to have cost just $500 to launch [Anon., 1985]. Many popular histories of the personal computer identify the introduction of *VisiCalc* as the critical event in the development of the industry. Thus, in Robert Slater's *Portraits in Silicon* [1987] we read:

> Suddenly it became obvious to businessmen that they had to have a personal computer: VisiCalc made it feasible to use one. No prior technical training was needed to use the spreadsheet program. Once, both hardware and software were for hobbyists, the personal computer a mysterious toy, used if anything for playing games. But after VisiCalc the computer was recognized as a crucial tool [pp. 265-266].

Chposky and Leonsis [1988] state in *Blue Magic*, their account of the development of the IBM personal computer:

> Then, in the Spring of 1979, a software program called VisiCalc was developed by two graduate students at the Harvard Business School.... For a full year, the program

> was marketed exclusively for the Apple II computer. This was a shrewd move on Apple's part because, in a number of instances, customers would go to a computer retail store to buy VisiCalc, and then ask for "something to run it on." The program was a deserved success, and it is often credited as the impetus behind the rise in Apple's revenues from $800,000 in 1977 to almost $48 million only two years later [pp. 7-8].

Finally, in what is easily the best and most restrained account of the development of the personal computer, Frieberger and Swaine's *Fire in the Valley* [1984], we read:

> VisiCalc was a novelty in computer software. Nothing like it existed on any computer, large or small.... Year after year, even as VisiCalc increased in price, the volume of sales rose dramatically. At first release in 1979 personal software shipped 500 copies per month. By 1981 it was shipping 12,000 [p. 230].

One can find similar accounts in virtually every history of the personal computer. Robert X. Cringley's delightful but idiosyncratic *Accidental Empires* [1993] has taken the process a stage further and made the "killer app[lication]" the central thesis of his book. *VisiCalc* has become an icon that overshadows the fact that there were three key generic applications that legitimated the personal computer: the word processor, the spreadsheet, and the database system. All of these applications existed in the corporate context (including the spreadsheet -- which as a financial analysis tool had existed in various manual and computerized forms since the 1930s).

The technical achievement of the early entrants into the industry was to write software using primitive techniques (often employing software tools that had long been abandoned in the mainstream software industry) and to shoe-horn the results into the tiny memories of personal computers. Equally important to their success was the attention to human factors that made working with a personal computer a comfortable and productive experience. The human-computer interface was at this time a very underdeveloped organizational capability in the existing software industry, giving a competitive advantage to those new entrants with backgrounds in developing computer games -- which required a highly developed, if intuitive, knowledge of human-computer interaction.

The late 1970s was a fiercely competitive period for personal computer applications software, with thousands of new entrants into the industry, almost all of which were undercapitalized, two- or three-person start-ups. By about 1980, however, a small number of market-leading packages had established themselves through superior marketing and the process of *de facto* standardization [Arthur, 1989]. After the launch of the IBM PC in 1981, the personal computer was fully legitimated as a serious corporate tool, and the associated hardware and software industries went into exponential growth. By 1983 personal computer software sales were approaching $1 billion, and a small number of clear industry leaders had emerged (Table 1).

Perhaps the most striking contrast with the 1970s packaged-software industry was the sheer volume of PC packages sold. For example, at the end of 1983 *WordStar* (MicroPro's top-selling word processor) had cumulative sales of 800,000, and the company was shipping 20,000 copies a month. Very few of the corporate software packages had cumulative sales in four figures. (For example, Informatics' *Mark IV*, reportedly the world's most successful software product, had just four thousand sites in 1983.) Another remarkable contrast with the traditional packaged-software industry was the importance of popular advertising in product promotion. One widely cited source put advertising costs at 35 percent of revenues, against 15 percent for R&D [Sigel, 1984, p. 127; U.S. Dept. of Commerce, 1984, p. 16; Myers, 1985, p. 83; ACARD, 1986, p. 52].

By about 1982 the gold-rush era was over, and three significant barriers to entry into the personal computer software business had been erected. The first was a technological barrier that resulted from the dramatically improving performance of personal computers, which were now approaching the power of a mainframe of the mid-1970s. The new generation of IBM-compatible computers that had begun to dominate the market were capable of running software comparable with that used on small mainframes, and similar technological resources were required for its development. (To take one example, whereas the original *VisiCalc* had contained about 10,000 instructions, mature versions of *Lotus 1-2-3* contained about 400,000 lines of code.) The second barrier to entry was the need for expertise about computer-human interaction. The sources of knowledge of how to create personal computer software with an attractive interface had become locked into the existing firms; such knowledge was not something that could be learned from the literature or in a computer science class. The third, and probably the greatest barrier to entry into the personal computer software business was access to distribution channels. In 1983 it was estimated that there were 35,000 products competing for a place among the two hundred that a typical computer store could stock -- for example, there were three hundred word-processing packages for the IBM-compatible PC alone [Sigel, 1984, p. 126]. A huge advertising campaign -- and therefore an injection of venture capital -- was needed to overcome this barrier.

One might expect that these barriers to entry would have resulted in a stable oligopoly of the existing personal computer software suppliers, but this was not the case. Of the top five firms and products listed in Table 1, only Ashton-Tate's *dBase* remained a market leader by 1990 (and Ashton-Tate itself had been acquired by Computer Associates); the other firms had become also-rans or gone out of business altogether. Today's four leading personal computer software firms -- Microsoft, Novell, Lotus, and WordPerfect -- were all minor players in 1983.

The reasons for this transformation are complex, but the dominant factor appears to be the importance of a single "hit" product, whose arrival can transform a balance sheet for several years, and whose demise can send the firm into a spiral of decline. Perhaps the most poignant of these dramatic turns of fortune was that of

Table 1. Cumulative Sales of Best-Selling Business Personal Computer Software, 1983

Program	**Publisher**	**Cumulative Sales (units)**
Wordstar	MicroPro	800,000
VisiCalc	VisiCorp	700,000
SuperCalc	Sorcim	350,000
PFS:File	Software Pub. Corp.	250,000
dBase II	Ashton-Tate	150,000
Total		**2,250,000**

Source: Sigel [1984, p. 125].

VisiCorp (the company that, as Software Arts, had developed *VisiCalc*). At its peak in 1983, VisiCorp had annual revenues of $40 million; by 1985 it had ceased to exist as an independent entity. VisiCorp was effectively wiped out by the arrival of a competing product, *Lotus 1-2-3*, compounded by a series of strategic errors.

The Lotus Development Corporation was founded by the entrepreneur Mitch Kapor in 1982 [Peters, 1985]. As a freelance developer for VisiCorp, Kapor already had the technological and human-factors expertise necessary to overcome the technical barriers to entry. Using his personal fortune of $1.7 million together with $3 million in venture capital, he developed a spreadsheet program that would compete with *VisiCalc*. The product, *Lotus 1-2-3*, cost some $1 million to develop and at the time of its launch was significantly ahead of the field in its technical capabilities and human-computer interface.

To overcome the most formidable barrier, marketing, Lotus reportedly spent $2.5 million on the initial launch of its new spreadsheet. Of the retail price of $495, about 40 percent went into advertising. With this blaze of publicity, *Lotus 1-2-3* sold 850,000 copies in its first eighteen months and instantly became the market leader. The launch of *Lotus 1-2-3* in 1983 was probably the last moment when it was possible to overcome at a stroke all the barriers to entry to launching a major new personal computer software product. There has been no comparable success in the last decade from a start-up.

The principal victim of Lotus' success was VisiCorp. In an attempt to capture a broader market, VisiCorp had launched a raft of new products -- a word processor, a database, and several other programs. It had also invested a reported $10 million in developing a "graphical user interface," *VisiOn*. All of these developments were funded by the revenue stream from *VisiCalc*. None of the new products succeeded against the market leaders, however, and after the launch of *Lotus 1-2-3*, revenues from *VisiCalc* declined rapidly. In 1985 VisiCorp merged with another company and ceased to have an independent existence [Sigel, 1985]. The VisiCorp-Lotus struggle was mirrored by several others, such as that between MicroPro and WordPerfect, whose positions were similarly reversed: in 1984 MicroPro's *WordStar* had a 23 percent market share, against *WordPerfect*'s 1 percent [Fertig, 1985, p. 164]; five years later, *WordPerfect* was hugely successful, and *WordStar*'s market share was essentially vestigial.

The biggest turnaround in fortunes in the personal computer software industry has been that of Microsoft. In 1980 -- when VisiCorp was a $40 million company -- Microsoft had an annual turnover of just $8 million and was almost unknown to computer users and the general public. At that time Microsoft had much more in common with a 1960s-style software contractor than with a 1980s personal computer software developer. Its co-founders, Bill Gates and Paul Allen, had developed a BASIC programming system in 1975 that had become the industry standard and that was shipped with Apple, Commodore, Tandy, and other microcomputers. In 1980 Microsoft obtained a contract to supply the operating-system software for the new IBM personal computer. The story of MS-DOS has been told even more often than that of *VisiCalc*; suffice it to say that Microsoft's real talent lay in being in the right place at the right time.

The IBM PC and its clones rapidly came to take nine-tenths of the personal computer market, with the user-friendly Macintosh the only serious competitor. Each IBM-compatible machine was shipped with a copy of MS-DOS, for which Microsoft received a royalty of between $10 and $50. With MS-DOS the standard operating system for the personal computer, Microsoft was assured of a constant revenue stream, which it used to broaden its product line, developing new products and acquiring others by take-over. It is worth noting that exactly the same strategy was adopted by the now-defunct VisiCorp. The strategy has been successful for Microsoft apparently because operating systems do not provide a good basis for competition: though people care passionately about their choice of a word processor or a spreadsheet, they do not very much care what operating system is used in their personal computer -- certainly not enough to switch to a rival product. Microsoft's revenue stream from MS-DOS set it apart from every other personal computer software supplier. Whether MS-DOS was good or bad was irrelevant: it was the infrastructure that supported the rest of the software industry, and (unlike IBM that originated the personal computer, and Intel that made the chips) its intellectual content could not legally be copied.

Throughout the 1980s, there has been intense competition to develop a successor to MS-DOS -- a user-friendly, Macintosh-style graphical user interface. As well as Microsoft, at least five major players were hoping to succeed in this market. All these attempts have been fraught. IBM, for example, spent a reported $1 billion in R&D, plus undisclosed advertising outlays, in developing its OS/2 operating system [Bakke, 1992]. Microsoft launched its *Windows* operating system on no less than three occasions (version 1 in 1985, version 2 in 1988, and version 3 in 1990), before meeting success. With *Windows* version 3, Microsoft appears to have a product hit that should see it well into the next century. But should *Windows* be eclipsed by a rival product, then Microsoft would lose its assured revenue stream, become reliant on its applications software, and be just as vulnerable to competition as any other player in the industry.

Conclusions

This paper opened by suggesting that a historical analysis of the software industry could illuminate a number of contemporary concerns. The first concern was the U.S. domination of the industry (primarily a concern outside the United States, of course). The likely prime reason for U.S. software supremacy is a paradoxical one -- government support for the industry. The paradox arises from the fact that, although the United States is non-interventionist in principle, in practice it promoted the early industry massively by creating a market for computers and software through programs such as the SAGE project, the Department of Defense's ADP program, and the NASA program, to mention only the largest [Flamm, 1987]. In Europe, though governments were interventionist in principle -- Britain, France, and Germany all had active policies to create thriving information-technology industries -- in practice their programs were always too small to have more than a marginal effect. The failure to develop computer hardware industries in the 1950s meant that the software industries could not follow in their wake in the 1960s.

The other reasons for U.S. dominance in the software industry stem mostly from early-start advantages. In the 1950s, the diffusion of computers in the United States was three to five years ahead of that in Europe, and the machines themselves were much more powerful. More powerful machines created a demand, for example, for programming languages such as FORTRAN and COBOL; the United States controlled the standardization of these languages, further consolidating the early-start advantage. This disparity was highlighted in 1962 by Christopher Strachey (a founder of the British software house CAP), who noted the huge gulf between the 200,000-instruction commercial programming-language translator that the Computer Sciences Corporation was developing for the Honeywell Corporation and an attempt by the United Kingdom's leading computer manufacturer to do the same thing "in a much smaller machine which in America would not even be regarded as an off-line printer. Clearly, there is something wrong" [Anon., 1962, p. 124]. With such an inadequate computing infrastructure, there was little opportunity for the U.K. software industry to develop competencies in constructing really large software artifacts.

There is currently great uncertainty about the nature of the threat from the software industries of Japan and the developing nations [Cusumano, 1991; UNIDO, 1993]. Historically, non-U.S. companies have done best where security considerations have protected them from American competition, or where local knowledge has differentiated them from the U.S. firms. For these reasons, European companies dominate the defense-software sector and compete well in niches such as financial systems and retailing, but do much less well in global sectors such as airline reservations systems. Japan, lacking a defense sector, has succeeded only in its largely protected mainframe software and in custom-software for Japanese companies [Feigenbaum, 1993]. This suggests that developing nations' software firms are unlikely to be a competitive threat to the West in software contracting, where indigenous knowledge of an industry is necessary. Moreover, at present, the emerging software industries compete largely in terms of raw code production; this is now less important than the human-computer interface technologies that are so critical to personal computer software.

Although the U.S. packaged-software industry benefited from early-start advantages, the size of the domestic market was a more crucial factor. The American market was at least twenty times larger than that of any other single country. For the packaged-software industry, recouping development costs over a large number of sales was the *sine qua non.* Whereas in the United States it was possible to obtain sales of most software packages in three or four figures, this was rarely possible in Europe. Moreover, the European markets were particularly attractive to U.S. suppliers because of the low marginal cost of software production. To sell packaged software required only a modest regional office and a sales force, and in the 1970s most of the major U.S. packaged-software firms developed successful marketing operations in Europe or sold under license in Japan.

The failure of the existing packaged-software firms to compete in the new market of PC software in the early 1980s is a major phenomenon that is not addressed by any of the official software industry reports. This historical survey of the industry suggests that the main reason for the failure was the nature of the firms' inheritance of organizational capabilities. This was most pronounced in the case of IBM, and was even a partial cause of its fall from grace in the early 1990s [Usselman, 1993]. IBM's traditional strengths in software were an understanding of business applications inherited from its office machine past and a sophisticated but bureaucratic organization geared to developing large software artifacts. IBM lacked a competence in consumer marketing, and its human-factors research was weak and oriented toward traditional computer usage. Thus IBM's software strengths were irrelevant to the development of personal computer software, while the competencies it lacked were imperative for success. What was true for IBM was true for all of the old-line computer manufacturers and software firms.

One might have expected that the gulf between the old-line packaged-software industry and the new PC software firms would have narrowed in time; but, although the technologies have converged, the cultural differences have not. For example, rather than join the trade association of the computer-services and software industries, ADAPSO, the new PC software firms established their own Software Publishers Association (SPA) in the mid-1980s. The board members of ADAPSO constantly pressured its president, Luanne James, to "take over" SPA, but she resisted because the difference in culture was "so blindingly obvious"; at an ADAPSO conference, for example, business dress is still the norm, whereas at an SPA conference, "if you show up wearing a tie they cut it off and throw you in the pool.... These are not compatible cultures" [James, 1995].

Perhaps the aspect of the software industry on which there has been least consensus among policy and technology experts is the role of R&D. In 1967-68 there was a major push from the computer science community to raise software production from a craft to an engineering discipline. The origin of this concern was a number of software disasters that hit the headlines in the late 1960s -- typically involving huge cost overruns or catastrophic system errors [Ceruzzi, 1989]. The result was to focus research on the need for better software production techniques and on the application of mathematical methods to develop error-free software. Most European public research funding, and much of the U.S. funding, has gone into improving software engineering technologies.

These policy decisions have often flown in the face of evidence suggesting the absence of any real link between R&D expenditure and industrial success. For example, one set of statistics for 1983 in the U.S. Department of Commerce report [1984, p. 31] showed that the traditional packaged-software firms ADR and Computer Associates were devoting, respectively, 13 and 12 percent of their turnovers to R&D, while the manifestly more successful Lotus Development Corporation was spending just 2.2 percent. One senses that beneath the surface calm of the official software-industry reports there lurked a polarization between the technologists, who tended to determine the practical implementation of research programs, and the policy analysts and economists, whose role was generally little more than advisory. Only occasionally does the trenchant view of a policy analyst emerge. One memorable example is Peter Grindley's report on *The U.K. Software Industry*, which devoted an entire chapter to condemning the software engineering initiative of the Alvey Programme, the U.K.'s main publicly funded information technology research program in the 1980s [Grindley, 1988; Oakley and Owen, 1989].

As a result of the technological fixation on software engineering, very much less attention was (and is) paid to human-factors research devoted to making software easy and attractive to use. The United States, however, was fortunate in having in the Advanced Projects Research Agency (ARPA) in the 1960s, a "man-computer symbiosis" program that fostered studies of human-computer interaction, at a time when the topic was barely recognized in the wider research community [Norberg and O'Neill, forthcoming]. Eventually these human-factors technologies diffused into a number of West Coast computer and software firms such as Xerox, Hewlett Packard, Apple Computer, and Microsoft. For all these firms, "usability" was not only a marketing requirement, but also a core technology -- and sophisticated techniques of requirements-gathering, product design, and evaluation were developed. Human-factors technologies are now locked into these firms, and it will be several years before the concepts are diffused sufficiently widely that they to no longer constitute a barrier to entry. Given this barrier -- and the trivial marginal cost of software production -- it seems highly unlikely that the U.S. dominance in personal computer software will be overcome in the foreseeable future.

Finally, a topic that none of the official software industry reports addressed is recreational software -- that is, software for computer games, personal finance and time management, and multimedia entertainment. Most of the reports dodge the issue entirely, although the U.S. Department of Commerce did at least recognize its commercial importance and dealt with it in a separate report [Watkins, 1984].

In 1982 domestic sales of U.S.-produced games software was estimated at $1.2 billion [Watkins, 1984, p. 20], which was more than 10 percent of the global software business. The other reports failed to discuss recreational software primarily because it was not perceived as software in the traditional sense: it was frivolous rather than serious; it was about ephemeral consumer spending rather than capital investment; it was typically created in an unstructured environment outside the industrial establishment of software engineering; and its cultural values were less concerned with functional requirements than with entertainment and graphic design.

Even today, recreational software is not recognized as a major sector of the industry. Thus Microsoft's insistence on selling a flight-simulator game in the late 1980s was seen more as an inexplicable whim of the company's founder than as a strategic position. Although very popular computer games such as *SimCity* and *Sonic the Hedgehog* are recognized as major cultural and economic phenomena, they do not feature in the computer trade press. Events are beginning force a change in this perception, however. For example, Microsoft's abortive $2.2 billion take-over of Intuit (the makers of *Quicken* personal finance software) has finally caused the computer press to take an interest in non-professional software. It helps, perhaps, that personal finance software sits precisely on the threshold between professional and non-professional computer usage.

There have been other signals that recreational software is crossing a threshold. Several mainstream software firms, including Microsoft, have created multimedia divisions and launched products -- typically reference works and educational software. Publishers and entertainment companies are also crossing over into electronic publishing -- the most dramatic example being DreamWorks, a reported $2 billion multimedia consortium headed by Steven Spielberg and others, with prominent investors who include Microsoft's Bill Gates and Paul Allen [Corliss, 1995]. Clearly, many of these enterprises have been formed in anticipation of a thriving market for information goods distributed via the Internet.

Whether or not these market expectations are met, the present-day recreational software business is of sufficient scale that it should be discussed alongside the traditional software industry. In the historical model of the software industry presented in this paper, recreational software can be seen as a culmination of two long-term trends that connect it to the three established sectors of custom programming, packaged software, and PC software.

The first trend is an ever-increasing emphasis on the cultural content of software as opposed to pure function. The first software artifacts in the 1950s were almost completely functional and mechanistic; typically, they converted data from one form into another (say from a programmer-oriented programming language into the computers' own binary code). With the introduction of more powerful computers and packaged software in the 1970s, however, program products began to incorporate a richer cultural component --

a payroll package, for example, would embody taxation rules and employment law, while an industry-specific package would codify many of the customs and practices of an industry. With PC software, the balance of functional and cultural content shifted still further -- in a spreadsheet program, for example, perhaps only one-tenth of a program was related to "number-crunching," with the rest supporting a user-friendly interface and facilities for data presentation and visualization. Recreational software can be seen simply as a further shift toward a richer cultural and artistic content.

The second long-term trend is the reshaping of the market for software products from a professional elite to the mass consumer; this shift has been accompanied by rising volumes and falling unit costs. In the 1950s and 1960s the price of a typical custom-written program ranged from $100,000 to $1 million, and the buyer of such an artifact would invariably have been located in the central computing department of an organization. The emergence of packaged software in the 1970s gradually took control away from centralized computer departments by enabling the user departments of a firm to participate in purchasing decisions. A 1970s program product, such as an industry-specific package or a database system, would typically have ranged in price from $10,000 to $100,000. With the arrival of the personal computer, software acquisition was made still more democratic. Individuals not only made their own software-purchasing decisions, but they often formed loyalty bonds with specific products; at the same time, the price of a PC package (typically in the range $200 to $500) made it affordable for a departmental, or even a personal, budget. Recreational software can be seen as another phase in this democratization of software, with computer games, non-professional finance software, and multimedia products all priced typically under $100. This is software for everyone.

References

Advisory Council for Applied Research and Development (ACARD), *Software: A Vital Key to UK Competitiveness* (London, 1986).

Anon., "Automatic Programming Languages for Business and Science," *Computer Journal*, 5 (1962), 105-139.

Anon., "Software: The New Driving Force," in Tom Forrester, *Introduction to Information Technology* (Oxford, 1985), 27-44.

Anon., "The New IT Industry Takes Shape," *Datamation*, 15 June 1993.

Armer, Paul, "SHARE: A Eulogy to Cooperative Effort," *Annals of the History of Computing*, 1 (1980), 122-129.

Arthur, Brian, "Competing Technologies, Increasing Returns, and Lock-In by Historical Events," *Economic Journal*, 99 (1989), 116-131.

Aspray, William, "An Annotated Bibliography of Secondary Sources on the History of Software," *Annals of the History of Computing*, 9 (1988), 291-343.

Bakke, John, "IBM's New OS/2 Software: It's A Go!" *THINK*, 58 (1992), 12-19.

Bashe, Charles J., Lyle R. Johnson, John H. Palmer, and Emerson W. Pugh, *IBM's Early Computers* (Cambridge, Mass., 1986).

Bauer, Walter, "Software Market's in the 70's," in Fred Gruenberger, *Expanding Use of Computers in the 70's: Markets, Needs, Technology* (Englewood Cliffs, N.J., 1971), 51-60.

Bauer, Walter, private communication to author, 18 April 1995.

Baum, Claude, *The System Builders: The Story of SDC* (Santa Monica, Calif., 1981).

Benington, Herbert D., "The Production of Large Computer Programs," *Annals of the History of Computing*, 5 (1983), 350-361. [Re SAGE software; originally published 1956.]

Brooks, Frederick P., Jr., *The Mythical Man-Month: Essays in Software Engineering* (Reading, Mass., 1975).

Burck, Gilbert, *The Computer Age and Its Potential for Management* (New York, 1965).

Campbell-Kelly, Martin, *ICL: A Business and Technical History* (Oxford, 1989).

Ceruzzi, Paul E., *Beyond the Limits: Flight Enters the Computer Age* (Cambridge, Mass., 1989). [See chap. 9, "Software," for an excellent historical analysis of software disaster stories.]

Chposky, James, and Ted Leonsis, *Blue Magic: The People, Power and Politics Behind the IBM Personal Computer* (New York, 1988).

Corliss, Richard, "Hey, Let's Put on a Show," *Time*, 145 (27 March 1995), 48-54.

Cringely, Robert X., *Accidental Empires: How the Boys of Silicon Valley Make Their Millions* (New York, 1993).

Cusumano, Michael A., *Japan's Software Factories: A Challenge to U.S. Management* (New York, 1991).

Feigenbaum, Edward A., *The Japanese Software Industry: Where's the Walkman?* (Stanford, Ca, 1993). [Video lecture, University Video Communication; no transcript known.]

Fertig, Robert T., *The Software Revolution: Trends, Players, Market Dynamics in Personal Computer Software* (New York, 1985).

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

Flamm, Kenneth, *Creating the Computer: Government, Industry, and High Technology* (Washington, D.C., 1988).

Flamm, Kenneth, *Targeting the Computer: Government Support and International Competition* (Washington, D.C., 1987).

Forman, R., *Fulfilling the Computer's Promise: The History of Informatics, 1962-198* (Woodland Hills, Calif., 1985).

Frieberger, Paul, and Michael Swaine, *Fire in the Valley: The Making of the Personal Computer* (Berkeley, Calif., 1984).

Gallagher, James D., *Management Information Systems and the Computer* (New York, 1961). [Case study on SABRE, 150-175.]

Grindley, Peter C., *The UK Software Industry: A Survey of the Industry and Evaluation of Policy* (London, 1988).

Harris, Brian, *BABS, BEACON and BOADICEA: A History of Computing in British Airways and Its Predecessor Airlines* (Basingstoke, Hampshire, 1993).

Ichbiah, Daniel, and Susan L. Knepper, *The Making of Microsoft* (Rocklin, Calif., 1991).

Imlay, John P., with Dennis Hamilton, *Jungle Rules: How To Be a Tiger in Business* (New York, 1994).

James, Luanne, private communication to author, 28 May 1995.

Kestin, Hesh, *Twenty-First-Century Management: The Revolutionary Strategies that Have Made Computer Associates a Multi-Billion Software Giant* (New York, 1992).

Kubie, Elmer C., "Recollection of the First Software Company," *Annals of the History of Computing*, 16 (1994), 65-71. [Re Computer Usage Company].

Lesourne, J., and R. Armand, "A Brief History of the First Decade of SEMA," *Annals of the History of Computing*, 13 (1991), 341-349.

McKenney, James L., with Duncan G. Copeland and Richard O. Mason, *Waves of Change: Business Evolution through Information Technology* (Boston, Mass., 1995).

Myers, Ware, "An Assessment of the Competitiveness of the United States Software Industry," *IEEE Computer* (March 1985), 81-92.

Norberg, Arthur L., and Judy E. O'Neill, *Transforming Computing: The Pentagon's Role, 1962-1987* (Baltimore, Md., forthcoming).

OECD, *Gaps in Technology: Electronic Computers* (Paris, 1969).

OECD, *Software, an Emerging Industry* (Paris, 1985).

Peters, Peter, "The Man Who Keeps the Bloom on Lotus," *Fortune* (10 June 1985), 92-95, 96, 98, 100.

Phister, Montgomery, Jr., *Data Processing: Technology and Economics* (Santa Monica, Calif., 1979).

Pugh, Emerson W., Lyle R. Johnson, and John H. Palmer, *IBM's 360 and Early 370 Systems* (Cambridge, Mass., 1991).

Rosen, Saul, ed., *Programming Systems and Languages* (New York, 1967).

Sigel, Efrem, "The Selling of Software," *Datamation*, 15 April 1984, 125-128.

Sigel, Efrem, "Alas Poor Visicorp," *Datamation,* 15 January 1985, 93-94, 96.

Slater, Robert, *Portraits in Silicon* (Cambridge, Mass., 1987).

United Nations Industrial Development Organization (UNIDO), *Software Industry: Current Trends and Implications for Developing Countries* (Vienna, 1993).

U.S. Department of Commerce, *A Competitive Assessment of the United States Software Industry* (Washington, D.C., 1984).

Usselman, Steven W., "IBM and its Imitators: Organizational Capabilities and the Emergence of the International Computer Industry," *Business and Econonic History*, 22 (Winter 1993), 1-35.

Voth, Ben, *A Piece of the Computer Pie* (Houston, Tx., 1974). [Re University Computer Company.]

Watkins, Ralph, *A Competitive Assessment of the U.S. Video Game Industry* (Washington, D.C., 1984).

Yates, JoAnne, "Application Software for Insurance in the 1960s and Early 1970s," *Business and Economic History*, 24 (Fall 1995), 123-34.