THE COMPUTER SOFTWARE INDUSTRY: PROSPECTS AND POLICY ISSUES

Peter Hall, Ann Markusen,
Richard Osborn, and Barbara Wachsman

Working Paper No. 410

July 1983

Institute of Urban and Regional Development

University of California, Berkeley

Table of Contents

THE COMPUTER SOFTWARE INDUSTRY: PROSPECTS AND POLICY ISSUES*

Peter Hall, Ann Markusen, Richard Osborn, and Barbara Wachsman

Of all the high-technology industries that have recently commanded national attention, the software industry is among the most promising. From less than $1 billion in the late 1970s, the industry's sales are expected to exceed $20 billion by 1990. Indeed, software will soon surpass the computer hardware industry in sales. Hardware receipts outran software by three to one in 1980, but by 1990, software sales will outpace computers by about 2.5 to one. Between 4,000 and 4,500 software firms now market over 10,000 products, and the average firm's sales are growing at a rate of more than 20% per year. This sector is highly labor-intensive, offering the prospect of substantial job creation. And, unlike the more speculative developments in biogenetics, its growth potential is an accepted fact. In the next decade, employment in the computer services industry in California alone has been projected to increase by 180%, and employment in companies specializing in software production could rise by 230% (Center for the Continuing Study of the California Economy, March 1982).

Our study was confined to California firms. California plays host

*This is a summary of a larger study titled "The California Software Industry: Problems and Prospects," done for the Commission on Industrial Innovation, State of California. The views expressed herein are those of the authors and not of the Commission or the Governor's Office.

to a substantial number of the more innovative software companies. Though California has only about 10% of the U.S. population, about 20% of the nation's employment in firms specializing in software products is located in the state (U.S. Department of Commerce). California's prominence in the software market is a result of its leadership in the related electronics and computer sectors, its generation of a highly-skilled professional/technical workforce, its innovative entrepreneurial climate, and its urban, cultural, and recreational amenities. But the software industry is rapidly changing. Its products and services are undergoing continual revolution, and its personnel needs are changing in a parallel fashion. Vigorous competition for California firms has emerged from centers in Texas, Washington State, and on the east coast, as well as from England and Japan. The internal dynamic of the industry, as well as the challenge from other regions and nations, raise questions about the industry's future both in California and in the nation.

We conducted the research in two phases. First, we reviewed the existing literature and data sources on the industry. The second and major portion of our research effort was dedicated to interviewing twenty-eight California software industry principals, including twenty-two firms, three academics, two private market research firms familiar with the industry and one state agency. The interviews permitted us to probe a number of features of industry structure and growth problems that were not adequately covered in the literature. A summary of the types of firms interviewed can be found in Figures A-1 and A-2 of the appendix.

We were thus able to identify three key issues for the future of the software industry. First, what will be the labor demands of this sector in the future and how can educational institutions respond to this challenge? Second, how can innovation and competition be encouraged in the industry? Third, what are prospects for any one region to retain and expand its software sector in the face of competition from other regions? We venture answers to each of these questions. We caution, however, that the software industry should not be considered a policy target in isolation from other sectors and from competing policy priorities.

A. History and Growth Prospects in Software

Software can be defined in the broadest sense to include programs or instructions that modify or run the computer hardware and extend its function beyond the actual computing; software maximizes the efficient and effective use of the equipment. Software, as distinct from coding or data entry, comprises all the activities associated with the successful development and operation of the computer system other than the hardware devices. The software industry includes (but is not limited to) products such as languages, programming tools, control programs, interpreters, monitors and supervisory systems, and professional and processing services. It can be delivered in a variety of mechanisms; it can be embedded in the hardware ("firmware") or as a separate product.

1. The Industry's History

Software has existed for almost thirty years, although little more than a decade ago the idea of selling software as the key vehicle of a computer system was unheard of. For an industry so new and important, it is curious how imprecisely its history has been documented, and how scarce authoritative information about its development is.[1] The first computers, strictly speaking, did not have "software", not until the early 1950s did John von Neumann develop the concept of the stored program: feeding the program instructions into the machine electronically, just as if they were data. With the successful implementation of FORTRAN in April 1957, the software era began.

Even after software became an entity in its own right, rather than simply an instruction set for the console operator's manipulation of the hardware, it was generally specific to its machine and was not written for use by any others than the specific operating technician. Even today, "standard" languages vary in minor but important ways; the FORTRAN or BASIC that runs on an IBM machine is not the same as the RATFOR or BASIC+ that runs on UNIX. Most importantly, software was not differentiated within the computer system by price; it was sold as a package with the computer hardware.

As computers were put to more commercial uses in the mid-1950s, data or service centers--the first software houses--were formed, and

---

[1] For some details of software development, see "Magic Moments in Software," Datamation, August 25, 1981, pp. 7-16, which also provides a bibliography.

contract programming firms developed. The first major customers of independent software vendors were the military, NASA, and the aircraft industry, who used the software in defense and space programs. A legal judgment against IBM in 1956, requiring the company to treat its service bureau as a subsidiary rather than as an internal activity, facilitated the development of an independent software products and service sector. Independent production began in earnest in the late 1950s. Such standard systems products as FORTRAN, BASIC, APL, and SAGE (an air-defense system language) were created by independent and, in many cases, individual programming efforts. User networks and information-sharing systems such as SHARE and GUIDE were also born in the 1950s (Datamation, 8/25/81, pp. 9-10). The final endorsement of this independent production and activity came in 1969, when IBM, under continued threat of antitrust action, was forced to revise its price structure and charge computer customers separately for application software, systems engineering, and training services--a process known as "unbundling."

Unbundling presented IBM with logistical problems, but was a tremendous benefit to the independent producers of software. It encouraged users to consider software as a product--a psychological point that signaled the start of the new software industry. Unbundling gave software independents credibility and made it easier for them to compete. New hardware competition also began, as manufacturers noted that unbundling offered vast potential revenues; they could shift their product lines into new areas, knowing that independent software producers would provide the needed software. The achievement of highly sophisticated hardware technology at falling costs has caused "logical complexity"--software, rather than hardware--to become the principal

constraint in the development or progressive improvement of computer systems.

2. Growth Prospects

So much computer machinery has been installed that the demand for software and software services is staggering. The recent development of the mini- and microcomputer systems has provided a second revolution in the computer industry and has led to the exponential increase in users and potential tasks; but there are troublesome aspects of mini- and microcomputing that have implications for the growth in use of computers. The usefulness of any microcomputer design depends on the software available; each personal computer manufacturer must have a wide range of programs available for its equipment for it to be successful. Software is clearly the pacing factor in the growth of personal computers.

Hardware technology has changed dramatically, but software remains a labor-intensive and therefore extremely costly process, and the rate of improvement in cost performance of software during the past 25 years has been very poor, much less than for hardware. Experts predict that in the future the relative cost of software will continue to escalate along the lines predicted in Figure 1.[2] With increased demand and continued cost increases, software is now the component of the computer system that largely determines its value: hardware has essentially

---

[2]Software costs will probably continue to exceed hardware costs as a portion of overall system costs for computers at the smaller end of the scale. But our interviews indicate that for larger systems, hardware costs will continue to be more that half of overall system costs.

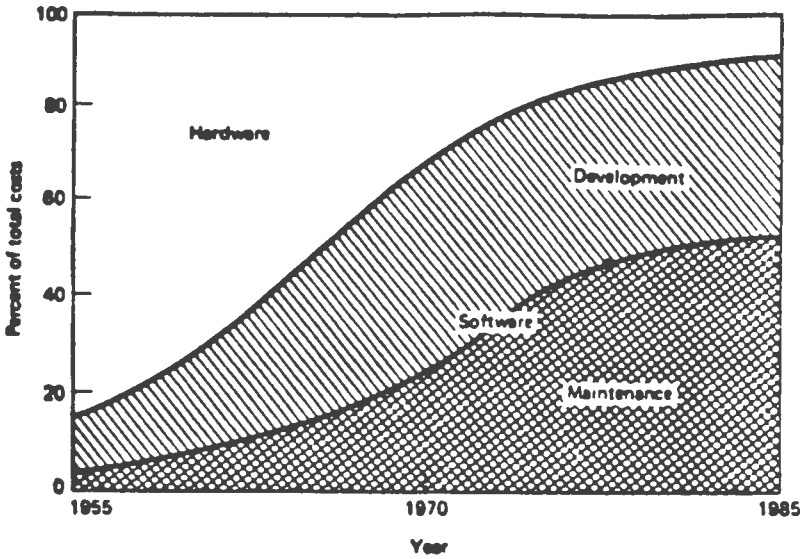


Figure 1.: Hardware/software cost trends

Source: Boehm, B.W., 1982, p.18

Figure 2: Comparison of Various Categories of Computers

Characteristics	Large Computer Systems	Medium-sized Computer Systems	Small Computer Systems	Minicomputer Systems	Microcomputer Systems
Purchase price ($)	1,000,000+	250,000-1,000,000	50,000-250,000	2,000-50,000 (basic form)	Several hundred -10,000
Main storage capacity	Extremely large	Large	Medium	Medium	Limited
Processing speed	Extremely fast	Fast	Moderate	Moderate	Slow
Number of I/O devices that may be supported	Extremely large	Large	Moderate	Moderate	Limited
Software support	Extensive	Considerable	Moderate	Moderate	Limited

Note: The relative terms "large," "fast," "moderate," etc. are included here to provide some general understanding of how computer systems differ.

Source: Stern, R.A. and Stern, N., 1982, p.14

become the packaging for software.

Although the numbers vary according to individual market research reports, the software industry today includes an estimated 4,000 to 4,500 independent software vendors, selling more than 10,000 products (New York Times, 1/8/81). The market includes a large number of medium-sized independent firms; some 2,700 of these individual firms recorded 1979 sales of less than $1 million (Fishman, 1982, p. 268). One analyst adds to that figure over 10,000 "mom-and-pop" programmers--professional consultants--in the Bay Area alone (San Francisco Chronicle, 10/18/81). Input, a Palo Alto consulting firm that follows the software industry closely, estimates that computer services revenues grew from $14.8 billion in 1980, with an overall growth rate of 24% (with individual niches growing faster than 40%), to over $22.1 billion in 1981. They predict that the market will grow by 140% by 1986 to $53 billion yearly (Business Week, 7/5/82, p. 80). The annual cost of software in the United States is now approximately 2% of gross national product and is expected to grow to 8.5% GNP by 1985 (Boehm, 1982, p. 17).

B. Computer and Software Technology

1. Computer Systems

Computer systems essentially consist of three elements:

Hardware. Usually thought of as "the computer," it is really the installation, network, or device that comprises an integrated group or computer system. This system can include a wide variety of

devices.

Software. These are the actual instructions or sets of programs, procedures, and related documentation of the computer system that enable it to function. This includes the direct instructions utilized by a non-technically oriented user in addition to extremely theoretical applications such as high-level languages and artificial intelligence, which have no commercial applications as yet.

Computer Systems. The relationship between hardware and software components that interact to satisfy the needs of users in an immense variety of applications.

Computer systems can be defined in terms of their computing and storage capabilities, which vary with size. Size categories range from mainframes (super, large and medium) to mini and microcomputers. Figure 2 provides an easy comparison of these categories.

a. Super or Giant computers have extremely sophisticated and large-scale capabilities beyond those of conventional large-scale computers. Speed increases approximately 10-12 times, as does the price. The traditional manufacturers of large-scale systems also produce the super computers.

b. Large mainframe computers ("mainframes") such as the IBM 370 have large memory capacity, in the million-byte (or "megabyte") range, and typically rent for over $100,000 per month or are purchased for several million dollars (Stern and Stern, 1982, p. 254). Large mainframes can take over a year to deliver, making accurate software very difficult to develop and deliver in a cost-efficient, timely fashion.

c. Medium-scale computers. These are widely utilized in business. They rent for an average of $10,000 per month and can have a purchase price of between $250,000 and $1,000,000 (Stern and Stern, 1982, p. 253). They can support a large number of input/output devices (utilized by businesses such as card validation services) and sophisticated software with a wide selection of languages or programs. IBM, Honeywell, Burroughs, and Sperry-UNIVAC are the leaders of this segment.

d. Minicomputers. The minicomputer system is difficult to define exactly, but in terms of the elements defined earlier, minis usually cost under $50,000 and have a memory size ranging from 32 to 256K; capacity can be increased by adding on integrated circuits (Stern and Stern, 1982, p. 258). Minicomputers have helped shape a rather different role for computers: the trend toward distributed data processing, which ties minicomputers and I/O devices (terminals, for example) together through an extensive communication network.

e. Microcomputers. "Professional personal computers," "personal office computers," "home computers," or "hobby computers" are the latest addition to the continuum of computing power, and are perhaps the most significant today in terms of software demand. They are distinctly unlike the larger systems described, although they have a broad range of depth and capabilities. They do not need specially designed rooms. They do not need professional staff; personal computers are designed to work at the individual level. They are relatively inexpensive (usually under $10,000). They are portable, usually as stand-alone desktop models that have no special electrical requirements. They are easy to use; the purchaser can install and operate on his own. Standing alone, however,

these smaller systems have limited memory capacity and less sophisticated software, and can support fewer peripherals. Add-on components, however, can improve and increase capacity (and cost).

Chiefly through entrepreneurial effort, software products and services have emerged as the fastest-growing segment of the computer industry. The rapid, innovative and highly fragmented behavior of the industry allows opportunities in the marketplace that are broad and extraordinarily diverse. Rapid growth has meant continual shifting of providers, products and market segments. Techniques and methods are rapidly replaced by new ideas and new products; producers and users shift in the market and add new roles; entirely new market segments appear with the creation of new applications. This rapid evolution of products and markets creates a problem for observers: the boundaries of software products and services become extremely difficult to define and measure.

2. Software Technology

As a marketable product, software takes the form of a magnetic disc or cassette on which is recorded a series of electronic signals. These magnetic tapes or discs contain the instructions that guide the hardware. The different types of software products fall into two major areas: systems software (sometimes called operating systems software) and applications software.

The physical process of the software design for a relatively simple program is based on a logical hierarchy of languages and systems that Bill Tuthill of UC Berkeley likens to the structure of an onion (see Figure 3). Layer 1 is the fundamental machine language, encoded in

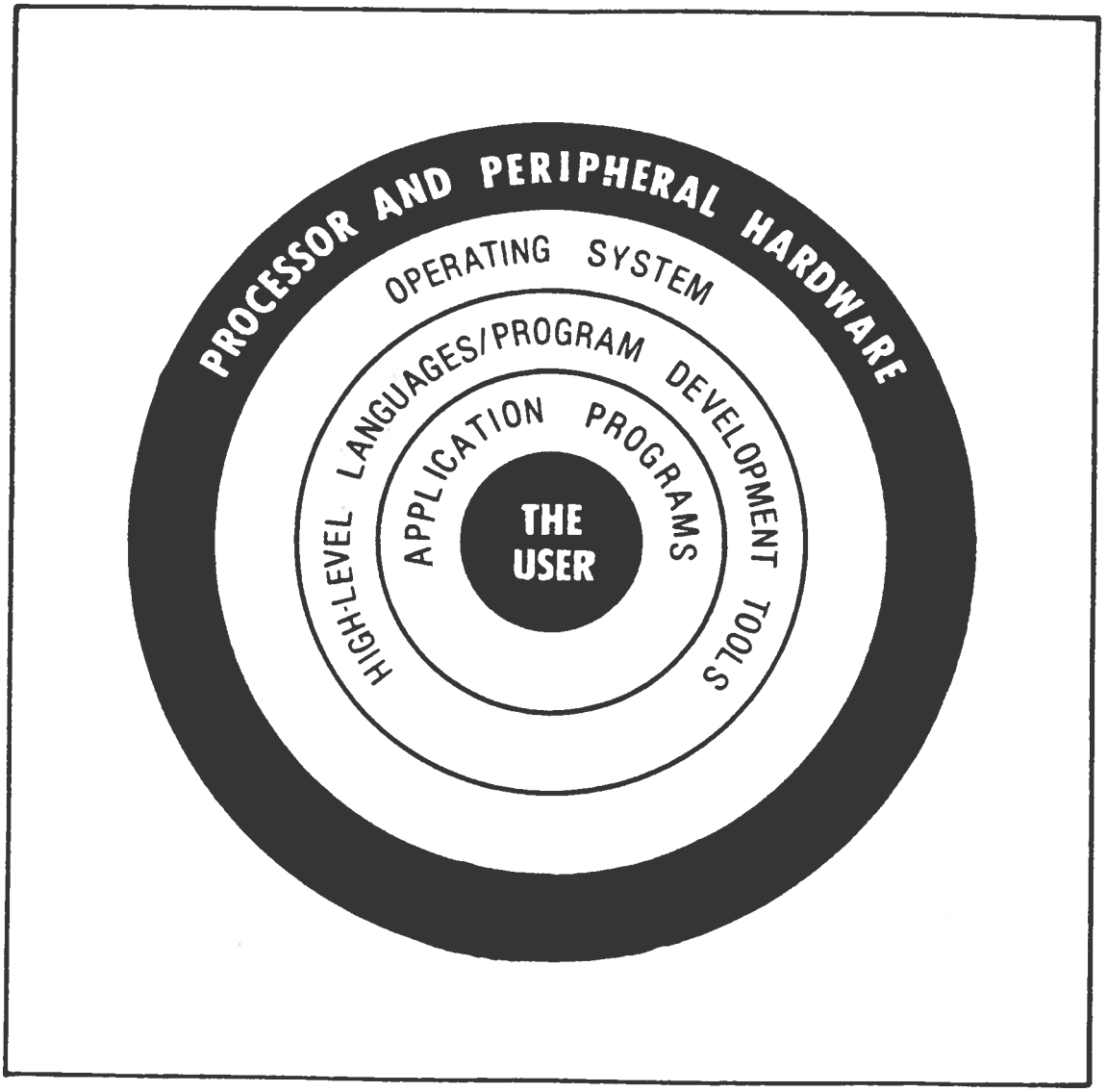


Figure 3: Layers of Computer Equipment

Source: Electronics May 21, 1981, p. 167.

binary digits: a digit is either on or off, either one or zero. Layer 2 is the assembly language where the words can act in place of the binary digits. Layer 3 is the operating system (such as UNIX). Layer 4 is the programming language (such as BASIC or PASCAL), and Layer 5 is the application package that provides answers to specific tasks like accounting problems or text processing (such as the programs VisiCalc and Wordstar). Layers 1 and 2 are considered part of the overall computer system and are usually provided by the hardware manufacturer. Layers 3, 4 and 5 are usually (but certainly not always) provided by software producers (independently or in a cooperative effort with the hardware manufacturer).

a. Systems Software

These programs run the computer system and enable the computer to perform the most basic operational functions. The overall system operates on application programs under the control of an operating system; and, generally, all programs other than the specific application programs are considered part of the operating system. As mentioned, systems software is often provided by the hardware manufacturer or leasor as part of the initial system, especially with larger systems. Systems software can be further classified to include the following:

i. System Operation Programs manage the resources of the computer during operation or program execution. They have names such as OS, DOS, CICS, and UNIX* (on which this report was run). These types of programs can

*UNIX is a Trademark of Bell Laboratories.

include data-base management systems (DBMS--which act as controllers for extremely large computer systems and improve the efficiency in the retrieval and processing of information), telecommunication monitors, etc. Prices for data-base management systems start at $7,000 (the low end is usually for use with minicomputers) and can exceed $100,000 when designed for a large mainframe. Prices are higher for custom-designed programs (Business Week, 9/1/80, p.53). These programs often come embedded in the hardware system in the form of a read-only memory (ROM).

ii. System Utilization Programs help manage the overall system operation more efficiently. These programs include job control and accounting routines, performance measurement programs, and scheduling systems.

iii. System Implementation Programs help prepare application programs for use on the system. These programs include assemblers, compilers, languages (multilevel), productivity aids, data dictionaries, interpreters, etc. We include both software languages (such as FORTRAN, PASCAL, BASIC), and high-level languages that enable programmers to develop application programs in the systems category, although some industry observers define languages as a third category of products, separate from systems and application software.

iv. The Supervisor routine, found primarily in larger systems, is the control or monitoring program of the operating system that eliminates the need for a human operator constantly to monitor such machine activities as start requests, restart job requests, maintenance of logs, etc. A special language called a job control language (JCL) is used for communication between human users and the supervisor.

b. Applications Software

This is the second major category of software products. In general terms, applications software consists of programs that perform specific functions or meet the specific need of the individual end user. They are created and written to perform a well-defined task for a particular application in a particular field, but, due to obvious constraints of economy, are not written to satisfy isolated single-user needs. More often an entire system or set of procedures, such as an accounts receivable system, will be designed and produced as a program. Application software responds directly to end user needs, and well designed application packages make the computer system as "user-friendly" as possible.

Application packages are also divided into two groups:

i. Cross-Industry or Horizontal packages, which perform applications common to many industries, such as accounting, manufacturing or sales management. VisiCorp in San Jose, for example, provides an extremely popular financial "spread sheet" program called VisiCalc, which can be used in a wide variety of businesses. Prices for these packages average $200-400; the median price of VisiCalc is $200.

ii. Industry-Specific packages, perform functions related to a specific industry, such as banking (demand deposit accounting), manufacturing (shop scheduling), or airlines (scheduling reservations). Argonaut in Oakland is one of many companies that produce, among other programs, specific accounting (general ledger, accounts payable, etc.) packages.

In contrast to the system software, which simply enables the computer to operate, applications programs are used by people involved in the daily operation of business or by individuals at home. Applications programs may be written by a number of different sources, including teams of in-house software programmers, independent consultants, individual entrepreneurs or the staff of hardware manufacturers.

c. Custom vs. Packaged Software

Software products can be provided primarily in two ways, either customized on an individual basis or presented as an off-the-shelf package. Originally, most software was built on a customized basis, but the trend is now clearly toward standard packages for both applications and system software. The latter rarely fit user requirements exactly, although increasingly they incorporate a wide range of options. Custom software takes time to produce, which makes it extremely expensive. The main advantage to packaged software, certainly, is price.

There are unique elements and problems in software design and development. Though it comes in physical form, software is a knowledge-based rather than a physical product, which means that the production costs are highly concentrated in the development stages. Compared to the initial research and development expenditures, the production costs of the actual software are negligible. Software does not physically wear out or depreciate, but it may be superseded by more efficient software.

Software design demands bright, highly skilled programmers and systems designers, of whom there is a marked shortage; it therefore remains

an expensive process. The ideal response to the shortage of skilled programmers would be to let the computer program itself, an event which may soon occur with the introduction of automated software development packages or program generators. These tools are applicable to any computer--from mainframe to micro--and for routine tasks they replace many of the programmer's functions.

C. The Structure of the Industry

An evaluation of competition in the software industry requires an analysis of differentiated market segments. While thousands of firms design and sell software, many of them operate only in very specialized segments.

1. Market Segments

For the purpose of this review, software services for all computer systems fall into three broad categories most commonly called Processing Services, Professional Services, and Software Products. These services and products can be provided on an individual customized basis, as a standard "off-the-shelf" package developed for many users, or as part of a complete "bundled" or "turnkey" system. Software products are the actual programs sold; they can be either customized or standard and traditionally fall into two categories: systems (operating systems) and applications programming. Processing services maintain their own computer facilities and use them to provide data processing and other services. Professional services offer the design of total computer systems, including a wide variety of special production and computing

services. "Turnkey" or integrated systems cut across all three of these industry segments and provide a customer with a self-sufficient computer system that includes as an integrated unit the hardware, software, and total support. Thus the user need not invest in systems design, applications development, or maintenance.

a. Processing Services: These firms provide their clients with data processing through batch services and remote-transaction computing services, plus provide other services such as systems management. The main benefit of a data processing services is that they offer access to large computing facilities. These processing companies can purchase and maintain their own hardware entirely or can offer timesharing services. Timesharing users or clients rent I/O devices (such as terminals), have access to the vendor's large computer system, and can lease their own software or buy software from the timeshare company. The clients then process the data themselves. This is different from batch processing or deferred processing, where users send data to the vendor's computer center and receive the output the next day. These services conveniently allow a business to computerize only part of its operation, such as payroll, without an extremely large investment. Time-sharing on a larger computer system is an extremely popular portion of this service segment.

More than 2,000 processing service firms produced close to $9 billion in revenues in 1980 (ADAPSO, 1982, p. 53). Processing services accounted for over 59% of the total 1980 revenues, and are projected to grow at a 16% annual rate through 1985 (Input, Inc., 1982). Remote processing was the largest service, providing 45% of this segment's revenues. Processing services allow companies to utilize computer services

without the extremely high investment cost of software development. The more specialized processing companies, such as those that provide service for one industry (such as attorneys or physicians), are currently the most successful because they have found individual niches and operate with a minimum of competition. The success of these firms, however, also depends heavily on the nature of the customer base and the ability of that base to weather the current recessionary climate. The fastest-growing source of income for this market segment is from services sold to the federal government, but this is still a relatively small (3%) portion of total processing services (ADAPSO, 1981, p. 55).

Recently, large corporations outside the computer industry such as Citibank, General Electric, and McDonnell-Douglas, and many national accounting firms, have entered this market, offering to sell excess data processing capability. General Electric recently announced computer service revenues of over $500 million, an 80% increase over 1980 estimates (Datamation, June 1982, p. 118). Many of these companies are now, however, facing antitrust litigation over these service offerings.

b. Professional Services: These firms design custom computing services for clients, complete with significant amounts of software as part of the service, offering an alternative to in-house programming. These consultants may also provide data processing management, special contract/custom programming, consulting services, or education and training. Professional services are provided by firms to clients who require custom-made computing systems and custom software. These are usually for particular tasks or functions for which standard products are not available.

Professional services firms produced $4.4 billion in revenues in 1981, which constituted 23% of the industry's 1981 revenues. A 17% growth rate is projected from 1981 to 1986. This particular market segment is characterized by a large number of individual, independent contractors. One estimate shows the market as consisting of approximately 2,000 vendors earning an average $250,000 in revenues each year (Software News, 12/7/81, p. 55). The number of vendors participating in this segment, and the estimated revenues, are extremely difficult to measure; under current definitions, any independent consultant who designs custom software may be considered to be providing professional services. Leading industry sectors utilizing these services in 1980 were the services and the banking and finance industries (ADAPSO, 1981, p. 78). A substantial amount of client contact is required in providing custom services, so these independent producers usually remain geographically close to their users.

The market for these services is based on the shortage of programming staff available to write individual software for business enterprises, and the resulting backlog of data-processing work. But the vendors of professional services themselves suffer from the programmer shortage, and lack software personnel with sufficient depth to produce complex new systems. Because this segment of computer services is the most labor-intensive, it is particularly affected by the shortage of skilled workers. Professional services firms have lately sought aggressively to attract new software professionals, which has only caused an even greater shortage of programmers in data processing departments, but reinforces the demand for outside professional services. Because of these high labor costs, and the steadily increasing use of standard

packages, custom professional services are not expected to be the most rapidly growing segment of the computer services industry.

c. Software Products: The market for software products is the most rapidly growing segment of the industry, with an estimated annual growth rate of over 42%. Market forecasts for software products show systems software as the leader in the market, with expenditures double those for application products (Dataguide, Spring 1981, p. II-1). Although the demand for application programs is rapidly growing, systems software will continue to lead this segment, with growth rates of over 30% annually. Input, Inc., estimates the market for system products growing from $1.5 billion in 1980 to $5.7 billion in 1985 (Dataguide, Spring 1981, p. II-1).

Software products firms derive 85% of their revenues from the sale of software, although professional services are also offered in conjunction with the sale. Large product firms obtain the majority of their revenue from systems software, while the small and medium-sized firms derive more income from applications software (ADAPSO, 1981, p. 63). Hardware manufacturers and larger independent software houses such as Boole & Babbage share the market. IBM is still the largest single vendor, and a large portion of systems software is still developed for IBM and IBM-compatible systems.

In contrast, hardware vendors have left the applications market to independent vendors, who supply over 75% of the current market of products (Dataguide, Spring 1981, p. II-1). This market is expected to grow from revenues of $800 million in 1980 to $2.6 billion by 1985. Analysts estimate that over 1,200 independents provide over 3,000

products in a rapidly changing marketplace where no one vendor commands more than 2% of the market (Dataguide, Spring 1981, p. II-3; San Francisco Chronicle, 10/18/81). Buyers in this market have an increasingly wide array of products to choose from.

The custom applications market also has unlikely entrants. Several large corporations, such as Westinghouse, Standard Oil, and Republic Steel, have attempted to recover their extremely high development costs for software by establishing software subsidiaries and attempting to market software in their industry-specific niches. This attempt has not been notably successful, simply because many of these large corporations do not understand the complex computer business. As a result, some corporations are now trying to re-sell their software through licensing arrangements with software companies (Business Week, 3/9/81, p. 74).

Applications products are usually targeted to industry-specialized areas, with nearly 60% of expenditures in the areas of banking, insurance, and manufacturing (Software News, 12/7/81, p. 55). Understandably, those business and industry sectors with the greatest number of computer installations have the largest percentage of expenditures for software products. The greatest growth is seen in business products, where the installation of micros and personal computers is increasing software expenditures rapidly. The increases in microcomputer sales are directly responsible for the tremendous success of software products such as VisiCalc and Wordstar (Datamation, June 1982, p. 118). Over 50% of all applications products are now obtained for micros or personal computers, and it is estimated that this market niche will grow at a rate of over 60% through 1985 (Software News, 12/7/81, p.

55). Many hardware manufacturers have entered this market because the availability of well-written application programs for microcomputers has been crucial to the successful market campaign of a small system. For example, Xerox had extremely sluggish sales on its initial foray into the personal computer market, because it provided only a limited range of software for its machines (The Economist, 2/20/82, p. 89). Already, these hardware firms are dominant in this market--Apple, Tandy, Hewlett-Packard, and Commodore. Apple Computers leads with 1981 revenues of $401.1 million, a 142% increase over 1980 (Forbes, 2/15/82, p. 113; Datamation, June 1982, p. 116).

Software product development is not subject to specific geographic location requirements. According to the recent ADAPSO survey, almost no software product firms serve only a local market. These firms need a very broad geographical market to support their products.

In summary, the past few years have seen a dramatic increase for processing services, professional services, and software products. Technical advances in personal computers, lowered costs, and increasing demand have opened vast new opportunities for application and system products. A new generation of computer-literate managers are aware of the capabilities of computers and are demanding new applications, especially in areas that must manage large data bases or areas that must respond to massive government reporting or tax requirements. The result has been the high growth of small software product companies, with a proliferation of system and application software to respond to growing demand. The larger hardware manufacturers are also entering this segment, and many industry experts predict a market consolidation.

d. Turnkey or Integrated Systems. Groups of companies across the three major segments just described form another rapidly-growing software market: the turnkey or complete integrated computer system, which includes hardware, software, service, and maintenance as a package. Three different types of companies usually function in this market: (1) specific turnkey vendors who purchase hardware and software from the original equipment manufacturers (OEM) and produce their own unique system; (2) hardware manufacturers who can license or produce their own software; and (3) data-processing companies who own or lease their own hardware or software. The price of the software component is the most rapidly increasing figure because of the continuing decline in hardware prices and the increased sophistication of users' demands for more applications. Turnkey vendors supply a number of fields, but narrow industry-specialized systems have been the most successful. Turnkey systems are used heavily in computer-aided design and manufacturing (Software News, 12/7/81).

The number of computer services firms selling integrated systems is projected to increase considerably. Processing services companies are extensively involved in this market; they sell the majority of these integrated systems in the banking and finance, medical, and governmental sectors (ADAPSO, 1981, p. 47). The success of this type of service has been slowed at times in the past by the problems associated with dependence on a particular hardware manufacturer, but overall they have achieved success by participating in every phase of the information-processing cycle and by taking advantage of the continually decreasing cost of hardware.

The current market shares of these three segments, and their expected growth rates, are shown in Figures 4. The smallest segment, software products, is expected to grow fastest in the early 1980s while the largest segment, processing services, will grow least rapidly. By 1985, the products groups will have doubled its share and control 25% of the market.

FIGURE 4

Revenue Growth by Market Segment

REVENUES	1980 billions	%	1985 billions	%	Average Annual Growth Rates
Software Products	2.6	12.4	8.4	24.9	42%
Professional Services	3.4	23.5	7.5	21.4	27%
Processing Services	8.8	59.1	18.8	53.7	17%

Source: ADAPSO Sixteenth Annual Survey of the Computer Services Industry, July 1981.

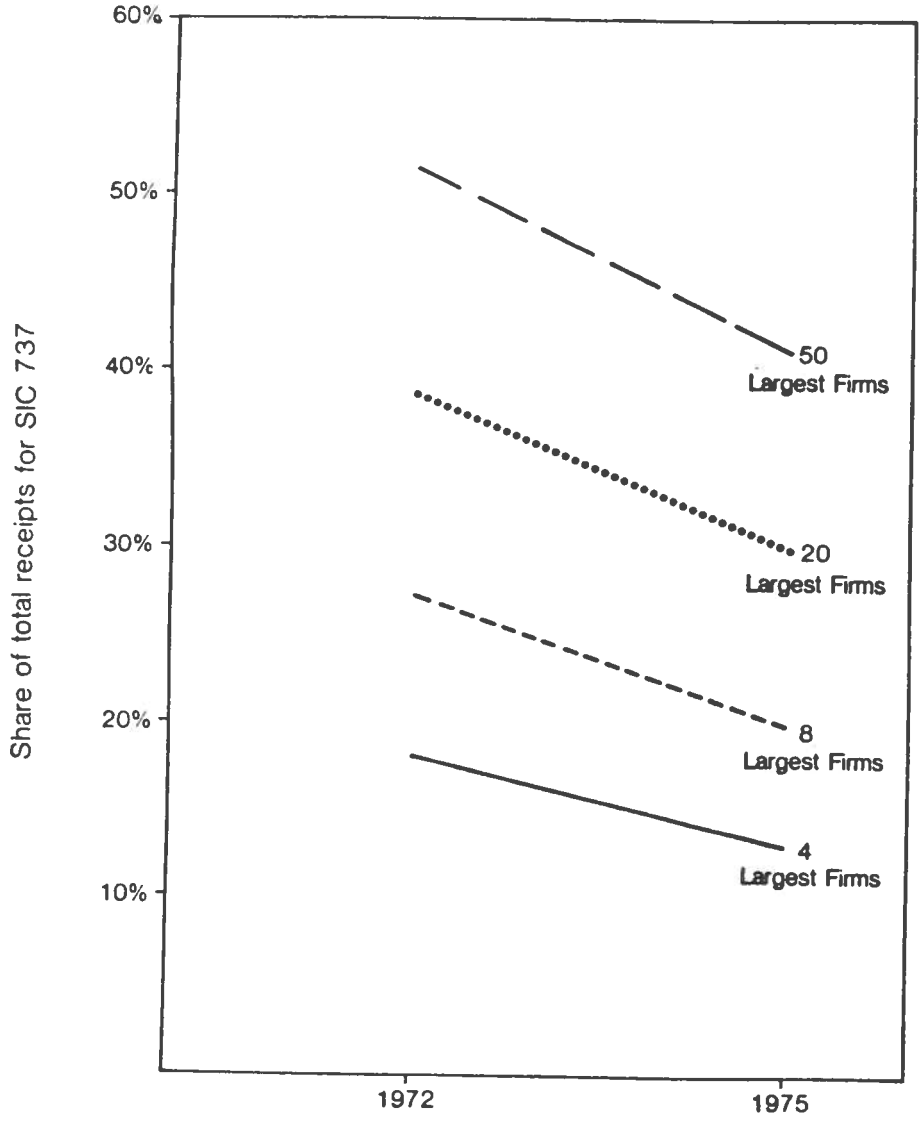
2. Concentration in Software

Concentration in the software sector is somewhat influenced by concentration in the hardware sector. Large hardware manufacturers are also five of the top major suppliers of software services. These five are IBM, Digital Equipment Corporation, Control Data Corporation, and

Sperry-Univac (Datamation, June 1982). IBM's software revenues are estimated at over $4.5 billion. DEC generated the second largest amount in software sales, with revenues of over $658 million (a 33% increase). Automatic Data Processing, Computer Sciences, and Electronic Data Systems are the top independent vendors of computer services only; ADP produced revenues in 1981 of $613 million, and Computer Sciences, which had a relatively small annual increase of only 11.4% because of heavy reliance on government contracts, produced revenues of over $624 million (Datamation, June 1982, p. 118 and Financial World, 11/15/81, p. 18). But even including these giants of software production, no single company commands more than five percent of market share in any area and the market remains extremely fragmented. This is distinctly unlike the hardware industry; the software market is not dominated by a single giant competitor, as hardware is by IBM. Figure 5 clearly demonstrates that in the 1970s, at least, the software sector of the computer industry has remained fragmented and highly volatile.

The "typical" software company is much smaller than these giants. It averages thirty to forty employees, with annual revenues of between $1 and 3 million. Annual growth is an average 29%, with profits before taxes at 11%, and average revenue per employee of $54,000 (Dataguide, Spring 1981, p. II-4). These small firms proliferate because entry into the marketplace is relatively easy for the independent computer services provider; compared to hardware manufacturing, little front-end investment is required. One person with one good idea can bring his/her idea to the marketplace. The software market has so many needs and demands that a small firm can easily carve a niche with a good product. Analysts estimate that 80% of the products of software goods and

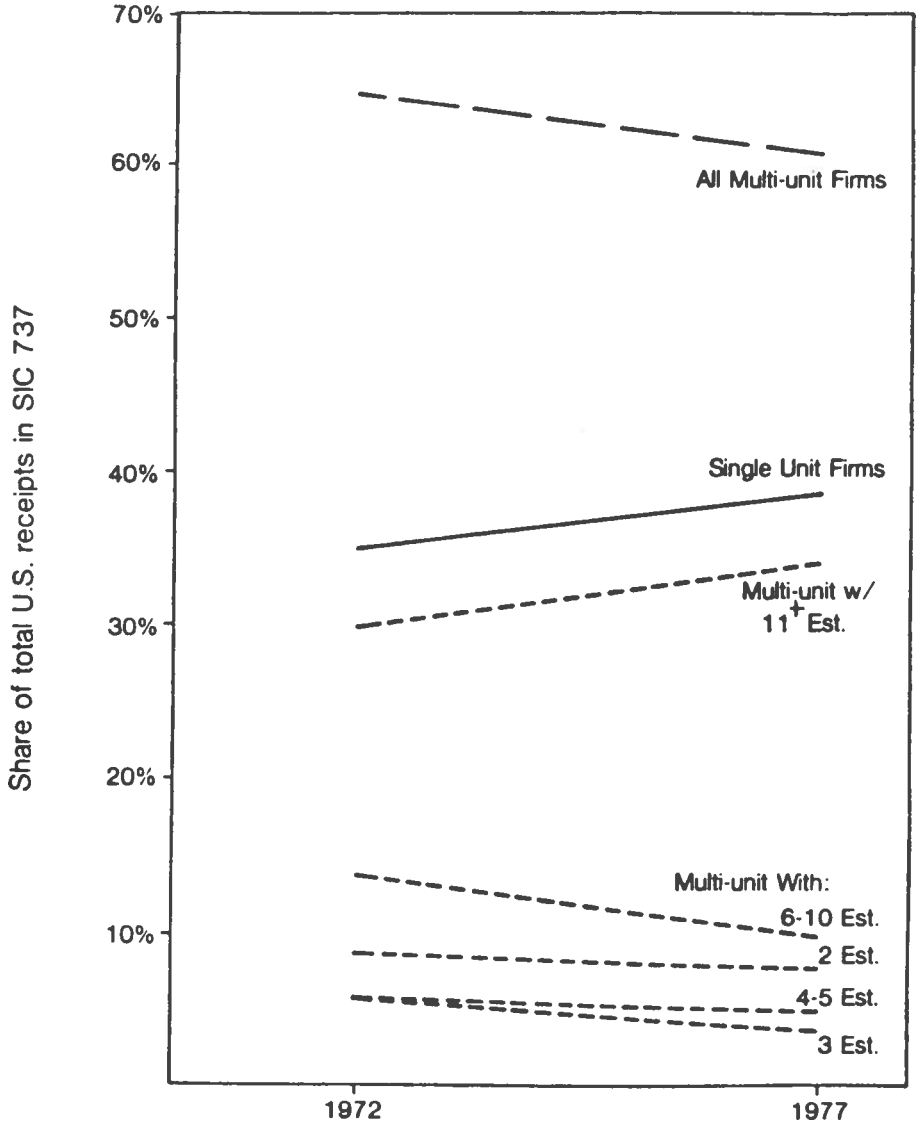
CONCENTRATION IN SIC 737 LARGEST FIRMS' SHARE OF TOTAL U.S. RECEIPTS, 1972-1977



Source: Census of Service Industries

Figure 6

SINGLE-UNIT AND MULTI-UNIT FIRMS' SHARE OF U.S. TOTAL RECEIPTS IN COMPUTER SERVICES (SIC 737), 1972-77



Source: Census of Service Industries

services comes from small independents (San Francisco Chronicle, 10/18/81, Section D, p. 1).

The data illustrated in Figure 6 support these assertions, but give some suggestion of future change. The share of all computer services receipts accounted for by single-unit firms was both substantial and rising during the 1970s, while the portion received by multi-unit firms with ten or fewer establishments dropped. But it could be an important indicator that the portion controlled by very large multiunit firms was quite high and also rose from 1972 to 1977. The industry could be bifurcating into one of very large and very small firms, which could result in an innovation-slowing dominance by the giants.

Effective competition, of course, can emerge from firms outside the U.S. The impact that Japanese competition has had in key areas of the computer hardware industry raises the question of whether software could be similarly affected. The visible successes of the Japanese have inspired considerable interest in their techniques for developing products and manufacturing systems. We asked most of our interviewees to assess the factors affecting software development in Japan and the competition that they might expect from Japan in their particular product areas. Thirteen of the twenty-five interviewees who responded to these questions seemed to feel that Japanese competition would not be significant for them. They cited the impediments in Japanese business practices and culture, (no venture capital, little entrepreneurial behavior, excessive perfectionism, etc.) and the fact that much software is specific to U.S. culture, e.g., accounting packages; the "cultural gap" and "language barriers". Twelve interviewees thought that the Japanese

will be significant competitors, especially in operating systems because these are least culture-specific. Several felt that the Japanese spend R&D money much more wisely than in the United States; that the Japanese have an education and training edge over the United States, and hence have more qualified people working in R&D. The Japanese were also described as being more user-oriented in software design and pricing.

3. Sources of Competition and Forces for Consolidation

Competition is affected by buyouts and vertical integration. More and more hardware manufacturers are trying to attract independently developed software, with enticements ranging from simple offers of directory listing, through royalty payments to extensive acquisition programs. And, increasingly, hardware vendors are simply acquiring software houses directly to meet the need for software. IBM, Digital Equipment Corporation, Xerox, and Texas Instruments are just a few of the large manufacturers that have very active acquisition activity (Mini-Micro Systems, January 1982, p. 115). Increasing acquisition activity will continue as hardware vendors attempt to deal with the shift in revenues from hardware to software. On the other hand, many software companies need the marketing potential and capital availability of a larger firm, but would like to remain independent and stay in the market (Mini-Micro Systems, May 1981, p. 23).

In addition to companies (including individual entrepreneurs) who write their own software and sell copies directly to customers in the marketplace--both systems and applications products, both customized and in package form--there are also middlemen who know the requirements of

hardware manufacturers and the marketing needs of many smaller independent software firms, and will market these software packages for one or more particular hardware systems. These companies, called software publishers, usually obtain the rights to sell programs written by individuals, provide duplication and distribution, and pay royalties to their individual authors. There are major advantages to selling software through these publishing houses: they have the resources to market and distribute products professionally--which includes the important tasks of packaging the software to be as "user-friendly" as possible, producing protected software, and providing the necessary user manuals and documentation (Korites, 1982, p. 153). Some software publishers specialize in one particular market segment or product, but most offer a full range of products from systems to applications. Another new entrant into the software distribution chain is the software broker, who acts as a middleman between the software writer and the software publisher or end user (Korites, 1982, p. 158).

In addition to the hardware manufacturers that support the production of software for their specific systems, many larger, independent software firms, in addition to selling their own products, also actively support the efforts of individual entrepreneurs who produce products compatible with theirs. Software companies that produce systems software very often provide support for vendors writing application programs that run on their systems software. Digital Research of Pacific Grove, for example, concentrates on selling a systems software package called CP/M. It actively encourages independent application producers to write programs using its language and systems software.

Hardware companies are also extensively involved in selling software. Many have their own in-house group of software producers and sell those programs with their hardware. Others, as we have noted, strongly endorse software publishers and use their products and licensing line to offer a catalog of software that will run on their hardware. Many microprocessor manufacturers, such as Datapoint, actually make design modifications in test hardware and decisions on architectural design based on which design has the most software available.

Microprocessor houses such as Intel are also actively encouraging outside software development efforts that are compatible with their chips, because the marketability is more favorable and production cost is lower if their chips are compatible with existing, reliable software. These firms are concerned because microcomputer manufacturers will not purchase chips unless a reliable operating system exists. High development costs are now forcing Intel to rely heavily on third-party software houses to supply software for its new 16-bit microprocessor, the 8086. They recently set up a "Software Distribution Operation" that will act as a software publisher, soliciting programs that will run on their chips (Electronics Review, 9/8/81, p. 39).

As technology rapidly changes, new application areas constantly appear, and the most successful individual firms will be those that can offer services to a specialized narrow industrial niche. They encounter virtually no competition, which means they can command higher prices with greater profit margins. As they move up the experience curve, they can then raise barriers to entry for other competitors. New marketing strategies will be demanded as the technology of computers improve and

the public's sophistication grows. For example, the market is increasingly demanding total computing service, as opposed to isolated products. Hardware and software firms are beginning to sell products, lease services, and act as consultants all at once. There will, however, be difficulties in expanding markets and distribution channels. Establishing multiple channels of distribution is very risky because it demands different sales personnel, new organizations, and new demands for capital. As an example, several software companies have attempted to expand by offering complete turnkey systems, only to find the capital requirements for purchase of the adequate hardware overwhelming.

The trend will clearly be toward vertical markets, where a vendor finds an industry niche and attempts to satisfy all the computing requirements: hardware, software, support, networks, and so on. Hardware companies are moving aggressively into the major service markets, providing entire packages for professions (like doctors and lawyers) and controlling production and distribution right to the retail outlet. Software companies like VisiCorp and Micropro are developing a "family" of products for vertical markets.

Merger activity to achieve vertical integration will increase. Major hardware companies are now moving to acquire successful software houses in order to add value to their hardware and to satisfy the new consumer demand for complete computing solutions. The meagre data available on acquisition activity endorse this. January through June 1981 saw fifty-seven acquisitions, valued at $244 million. In 1980, this same period witnessed forty-two mergers, valued at $164 million (New York Times, 11/8/81, p. 28). The structure of successful companies

will shift from free-wheeling, engineering-oriented, entrepreneurial ventures to established firms run by Business-School-educated marketing and financial strategists in order to exploit the increasingly competitive software marketplace. There may be a shift from multiple small, independent companies to a consolidated industry of a few conglomerates. But while there is clearly a shift toward consolidation, there is still lively activity among independents and large numbers of individual entrants into the industry.

D. Job Creation, Occupational Structure and Labor Force Needs

1. Job Growth

It is difficult to bring together complete data on software employment (or on the other indicators used in these sections), because software production is not limited to independent software firms, that is, software firms that produce only software. The Standard Industrial Classification (SIC) groups substantially involved in software production include the following:

SIC 737: Computer and Data Processing Services

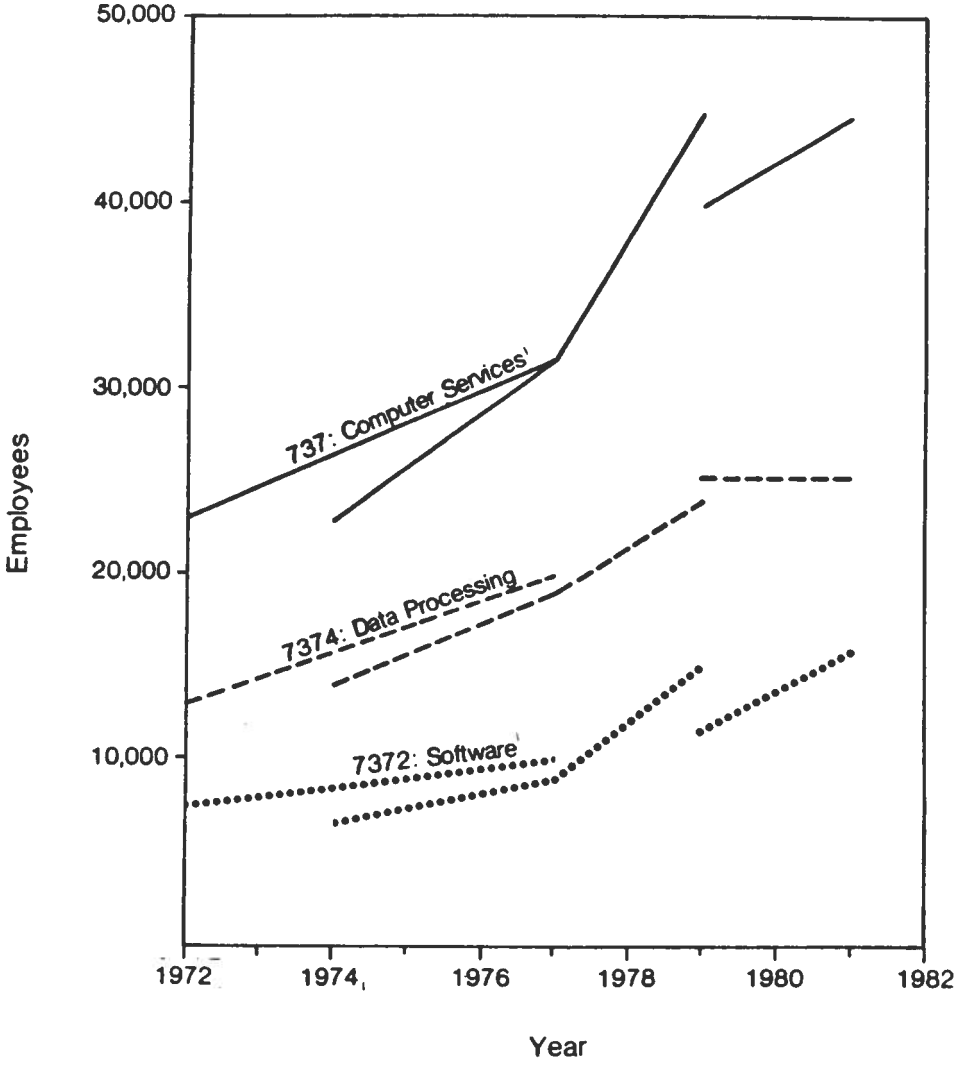
- SIC 7372: Computer Programming and Other Software Services
- SIC 7374: Data Processing Services
- SIC 7379: Computer Related Services, not elsewhere classified

SIC 357: Computers

Figure 7 shows an estimate of all California employment related to software production. The projection begins with 1970, because 1972 is

Figure 8

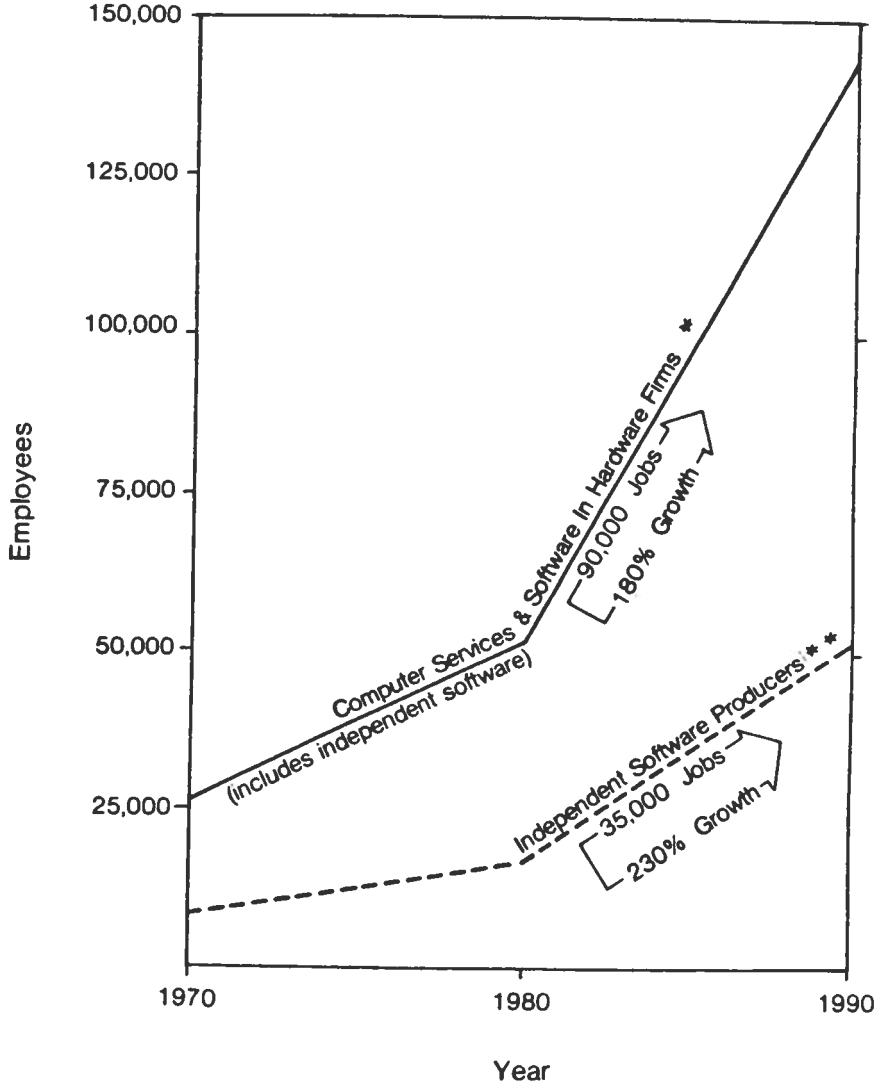
CALIFORNIA EMPLOYMENT IN COMPUTER SERVICES, 1972-81



Sources: 1972-77 Census
1974-77-79 County Business Patterns
1979-81 State of Calif., Emp. Dev. Dept.

Figure 7

REVISED EMPLOYMENT PROJECTIONS FOR CALIFORNIA: COMPUTER SERVICES & SOFTWARE IN HARDWARE FIRMS, INDEPENDENT SOFTWARE PRODUCERS 1970-1990



* SIC 737 + 10% of SIC 357 ** SIC 7372

Source: Center for the continuing study of the California Economy
are our estimates

the first year that SIC 737, Computer and Data Processing Services, was separated from SIC 7392, Business Consulting Services. The estimate through 1980 is supported by available data (see Figure 8). The projection through 1990 should be treated with great caution since there are reasons to believe that personnel requirements in software production could undergo major changes, such that employment requirements could level off or drop.

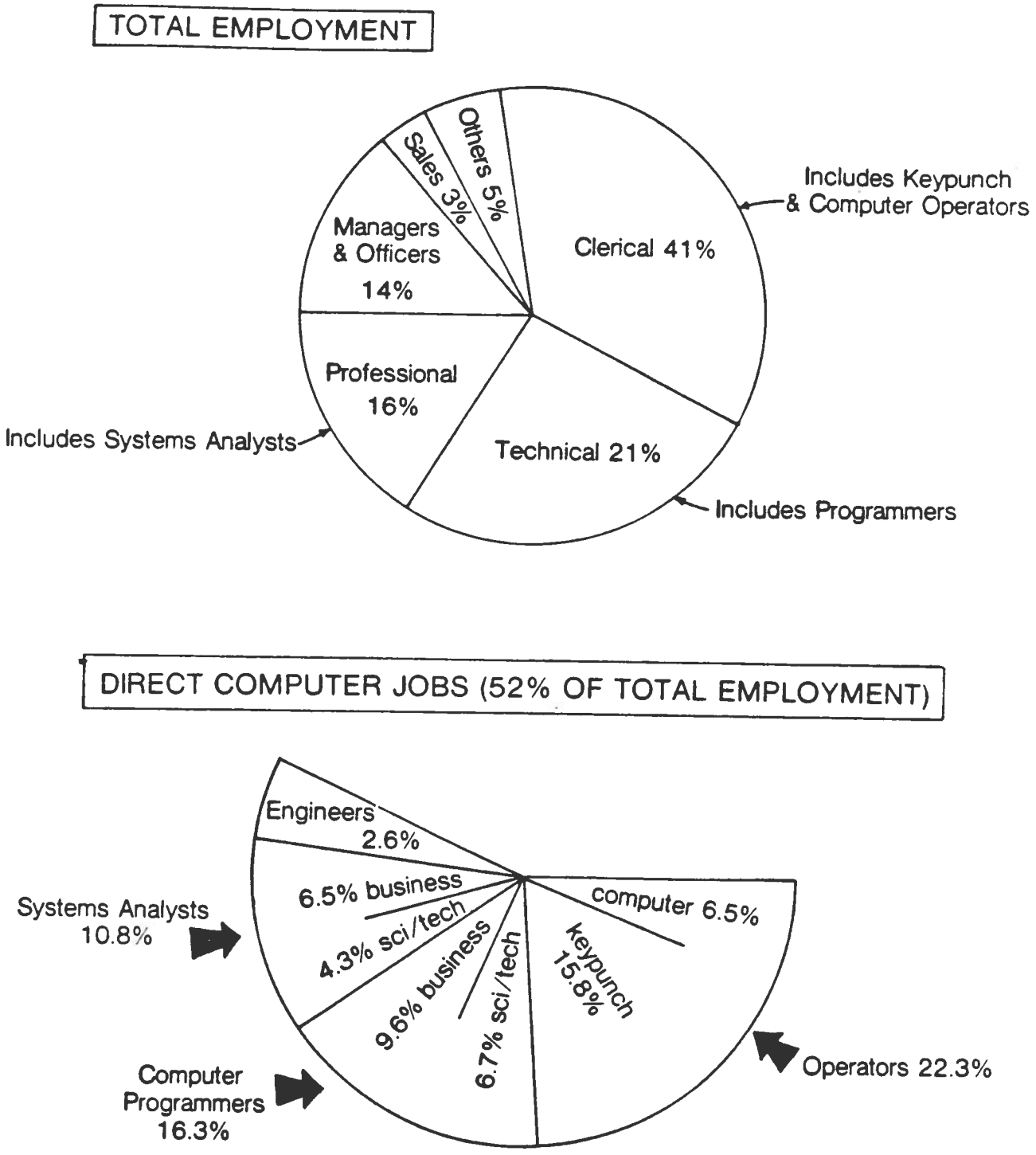
2. Occupational Structure

Software production requires a larger number of managerial, professional, and technical workers than most industries. For example, in national averages, the computer services sector (SIC 737) has a smaller portion of non-supervisory workers than the service industries as a whole. A detailed breakdown of the occupational distribution of the computer services sector in California is given in Figure 9. It shows that on average, about half of the employees in this sector work directly with computers. But few skills are required in the 'operator' category, so in effect only about thirty percent of the employees in the sector are skilled computer specialists.

Compared to manufacturing (see Figure 10), the computer services sector in California has a very high portion of employees in management, professional, and technical occupations: 51% compared to 19%. And it is clear that the real "production" workers in computer services are those in the clerical workers category. This means that only about a third of the spectacular growth of employment in computer services is in well-paid skilled occupations with a relatively secure future, but the other

Figure 9

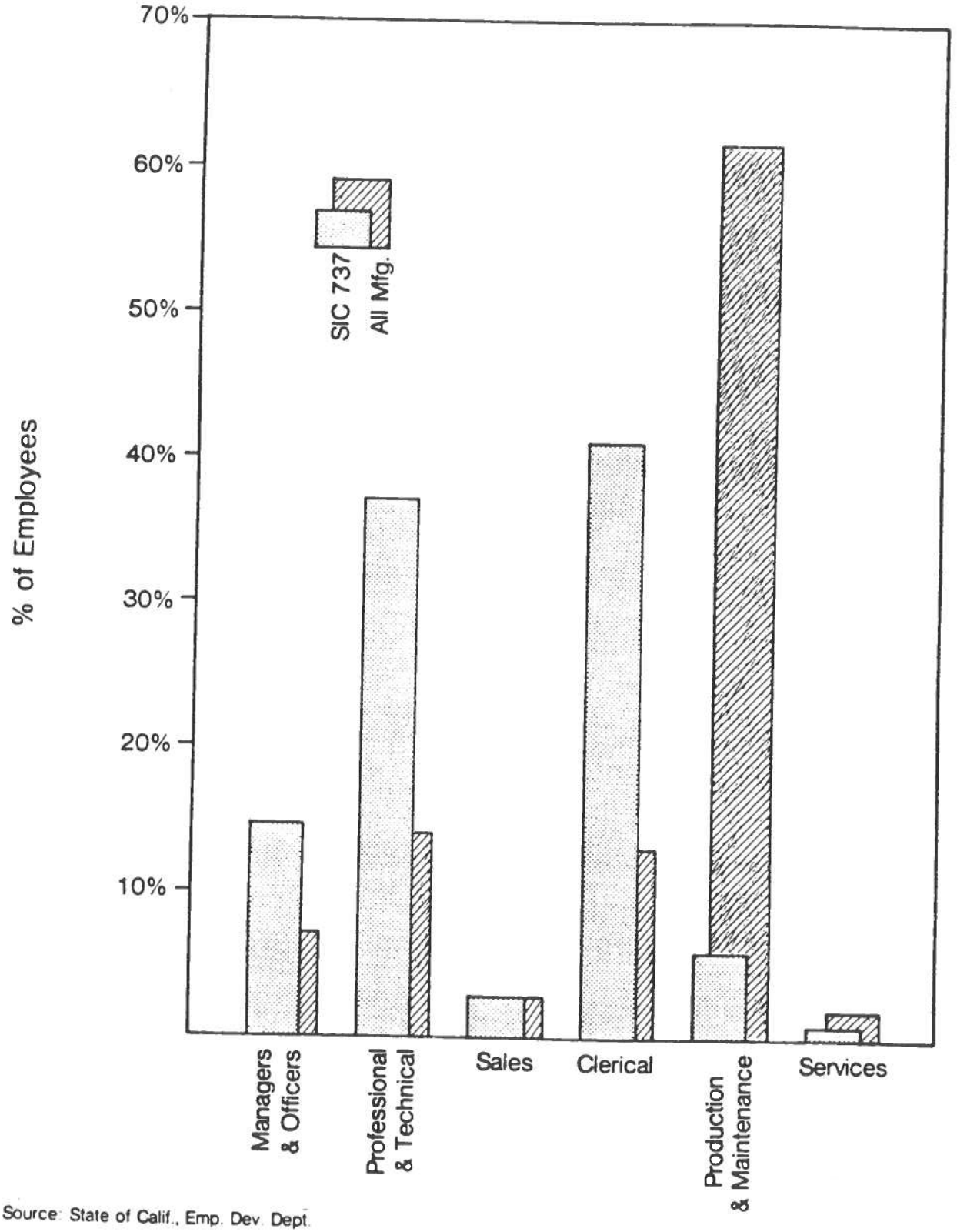
EMPLOYMENT BY OCCUPATION IN COMPUTER SERVICES (SIC 737), CALIFORNIA, 1978



Source: State of Calif., Emp. Dev. Dept.

Figure 10

OCCUPATIONAL STRUCTURE OF COMPUTER SERVICES COMPARED TO ALL MANUFACTURING, CALIFORNIA 1978



Source: State of Calif., Emp. Dev. Dept.

two-thirds of the jobs are relatively low-paying, less skilled and less secure.

3. Productivity and Incentives

Experts say the software needs for the microcomputer and microprocessor segments of the hardware industry will be hit hardest by labor shortages. The President of Intel predicts that unless radical new productivity tools are developed, over 1 million software engineers will be required over the next decade to service the micro market alone (Electronics, 5/8/80, p. 43). Experts in 1980 estimated that programmer demand already outstripped supply by at least 50,000 (Business Week, 7/1/80, p. 47).

In addition to acquisition activity, other approaches are used by individual firms to address the shortage of skilled labor. Companies that are in a well-capitalized position are attracting people through increased salaries and benefits. These companies believe the economic system will solve the shortage of programmers: more people will turn to programming as a career as it offers bigger salaries.

Other companies are doing everything possible to get greater output for the same amount of intellectual effort--and this includes substantial commitment to productivity tools to facilitate language processing, as mentioned above. They include automatic programming generators (which make programs more readily understood and transferred among programmers), text editors, and debugging devices that make correcting programs much easier. Other inventions include new languages that make it easier to write programs and will someday allow non-technical users to

write their own software. Computer-aided design systems are also being used as simulation design tools. This allows software to be produced and debugged as soon as the hardware is finished. Hardware manufacturers are also doing whatever they can to reduce dependency on human labor in programming. They are incorporating more self-diagnostic capabilities into machines, such as embedded "application chips" that can do much of the work previously done by systems programs.

Changes in job structure and labor markets are also emerging to bridge the labor gap. Companies such as Control Data Corporation are experimenting with entirely new job structures. They are training "software paraprofessionals" in an attempt to break down the development process and use lower-paid, less skilled technicians to do the routine work (Electronics, 5/8/80, p. 143). To increase the efficiency of the software labor market, specialized personnel firms (known to the personnel they seek as "head-hunters" or "body-snatchers") are springing up to orchestrate the limited amount of software manpower available. Several firms now lend custom programmers on a contract basis for very lucrative amounts. Another fairly controversial idea is the "software factory," which divides the programming tasks as in the manufacturing industry (see below). The imposition of this type of structure, however, has not been particularly successful. Even with these various attempts to "break the software bottleneck," experts such as M. L. Dertouzos of M.I.T. predict "constant costs and increasing opportunity for trouble" (Business Week, 9/1/80, p. 53).

4. Labor Force Requirements

The present demand for university-trained computer scientists to design software is obvious. But other job training and educational implications are not obvious. What other levels and/or fields of education are desirable for software designer jobs? What are the employment opportunities for minorities and women? Is the software industry the answer to plant closings in other industries? What kind of education will prepare people for the software design jobs of 1990 and 2001? From our interviews, we arrived at tentative answers to these questions.

a. Current Demand for Software Designers

Though there may be a fairly specific set of aptitudes required to design good software, there is quite a bit of variety, at present, in the educational background that programmers have and the kinds of work that programmers do. Figure 11 suggests a framework for describing the variety of programmers and programming tasks.

i. Education and Training

Most interviews indicated that the greatest need is for "supertechs": systems programmers and architects, most of whom have Master's degrees or Ph.D.s in Computer Science or Electrical Engineering. For the less-than-supertech level, nine interviewees agreed (and none disagreed) that aptitude plus experience were acceptable or preferable substitutes for academic degrees in Computer Science. In one firm 40% of the programmers have no Bachelor's degree, and a large percentage of the programmers for the State of California have less than four years of

Figure 11

PROGRAMMER BESTIARY

	Common Name	Tasks	Computer Knowledge Required	Most Common Academic Degrees	Ability & Experience Substitute for Degree?	Salary Range
TECHNICIANS ▢ SUPERTECHS	Programmer	Maintain existing programs, write parts of applications programs	One programming language	Less than a BS in CS ⇩	Yes	$15,000-30,000
	Programmer, Programmer/ Analyst	Structure and write applications programs	Programming language(s), Operating systems(s)	Bachelor's in other subjects, BS in CS, MBA ⇩	Yes but less common	$20,000-40,000
	Systems Programmer	Structure systems of applications packages, design operating systems	Programming languages, operating systems, assembly languages, some knowledge of hardware	BS, MS, PhD in CS, EE, math ⇩	Possible but not common	$30,000-60,000
		Design integrated hardware and software systems	All of the above, plus a thorough knowledge of hardware	MS, PhD in CS, EE	Not likely	$40,000+

college. In most firms there are several "self-made" types--including some legendary programmers who have backgrounds as musicians, Ph.D.s in French History, and so forth. Yet in a somewhat contradictory response, seventeen of the interviewees said that they generally look for a minimum of a B.S. in Computer Sciences, Electrical Engineering, or Mathematics when hiring programmers.

ii. Women and Minorities

In many firms, all or nearly all programmers are white males. In few firms did it seem likely that conscious selectivity produced this homogeneity. Certainly the pool of available programmers, particularly of those with advanced degrees, is dominated by white males. Thus while popular impressions of programming cast it as a mixed gender occupation, our findings suggest that in the more creative software design profession (as distinct from routine programming in large institutions), men predominate.

But we did notice a tendency for the presence of women as programmers in a firm to be related to the presence of women in top management in that firm. While this observation does not justify strong conclusions--statistical or otherwise--it suggests that workforce composition can be shaped by management to a certain extent. It seems that this constitutes a strong argument for maintaining diversity among firms in the software industry, and for continuing to encourage affirmative-action programs. A related issue, unexplored by this study, is the degree of discrimination against women and minorities at secondary and university levels.

iii. Is an Employment Shift from Declining Industries to Software Possible?

By any of a variety of measures present-day employment growth in computer/information industries in general, and in the software sector in particular, is colossal. One of our sources related to us the estimate that if current growth rates continue, 20% of the U.S. labor force would be programmers by the year 2001. But do estimates like this give a real basis for hoping that software growth could keep unemployment down? Our interviews gave us several reasons to be cautious about such assertions.

First, as we have tried to show, present-day programmers are basically a professional class with a fairly specialized set of aptitudes and even specialized education. That implies that only a small portion of the people who can no longer expect to be employed in declining industries such as manufacturing could be redirected to software production as it is now structured.

At most, there is a modest employment demand that can be filled by "paraprofessionals" who have less than four years of college, provided that they have the special aptitudes required for programming. And there is some chance that changes in the structure of software production (see the next section in this chapter) could result in an increased demand for such paraprofessionals.

Furthermore, indirect demand for employees in software firms to support the direct software producers should not be overlooked. In firms for which we have estimates, 30-80% of the employees had non-

programming jobs. And given that most observers believe that the growth of the software industry will lead to the growth of several related industry sectors, the employment multipliers associated with direct programming employment may be quite high. On the other hand, computerization may destroy jobs elsewhere. And in the long-run demand for programmers must be considered, to which we now turn.

b. Future Demand for Software Designers

Information processing is a derived demand. Hence it seems reasonable to expect that there will be upper limits on the place that computer industries will occupy in an economy. In even the most complex, and hence most computerizable, industrial economies, the information-processing industries could level off at 4-5% of GNP or less (in the United States, we were told, they are now at about 2.5%). At present, demand for programmers and other software designers is far in excess of supply; nearly all software specialists are highly skilled and are seen as professionals and creative individuals rather than as production workers in an industrial hierarchy. As a result, software costs are dominated by high personnel costs. While computer hardware costs are dropping sharply, software development costs have been rising. The shortage of programmers places noticeable limits on the growth of the software industry, which, in turn, imposes limits on the growth of hardware sales.

All of this adds up to a strong drive to make programming more efficient, that is, to reduce the number of people required to produce the same number of units of software and/or to transfer as much work as

possible from highly trained (expensive) to lesser trained (cheap) personnel. One model for this is the "software factory," a hierarchical, assembly-line in which software production is broken down into small stepwise tasks which are performed by less-skilled programmers passing their little pieces from cubicle to cubicle. None of our interviewees seemed to think that this model will be very successful, because software development is not easily routinized, and because individual programmers have too much influence on the overall project, even if they are given only a small part of it to work on.

The code generator (or program generator) model of the future software development process seems to us more likely to bring about significant productivity gains and help reduce the problems of unfilled need for skilled programmers. Seven of our interviewees think (and none disagree) that code generators will be developed. Use of code generators might require progressively fewer specialized skills. All the arcane incantations of Pascal, PL1, JCL, LISP, and even SpeakEasy and the like may become unnecessary, as communications with the computer come to resemble conversational English. Code generators will require more hardware, but as hardware becomes cheaper and programmers become more expensive, a familiar process of substituting machinery for labor seems quite likely to occur.

But there is not much certainty in projecting what exact effects the development of code generators will have on demand for "programmers," or the kinds of skills one will need to be whatever the future equivalent of a "programmer" turns out to be. Future computer systems may be designed so that, with the use of such tools as code generators,

most programming will be structured by end users who are not computer specialists, and who communicate with the computer in something like conversational English. Such systems would be designed by supertechs, Ph.D. types, to minimize the need for programmers as intermediaries. Supplementing such supertech-designed, user-friendly systems would be larger systems which themselves design user-oriented subsystems. The supertechs who designed the designing resource having perhaps long since moved on, "the resource" may function fairly independently, needing only occasional debugging and maintenance of itself and its offspring--a kind of computer family psychiatry which itself would require supertech status to administer.

This trend should have two effects on software employment in the next five to twenty years. First, it will cause demand for less-than-supertech programmers to level or drop. Second, it will fundamentally change the tasks and skills expected of programmers. "Programming" in future computer systems could involve four categories of employees:

1. The supertech, who designs and maintains nearly self-sufficient hardware/software systems.

2. The end user, whose main job is to do something else besides work with computers.

3. Professional intermediaries who specialize in using computers in problem solving.

4. Paraprofessionals who do lesser skilled programming tasks.

Employment demand for supertechs is sure to be high, and they will probably be required to have considerable specialized computer education. The employment and education demands for the end-user category will depend on the rest of the economy. Employment demand and education requirements for programming professionals or paraprofessionals is

highly uncertain, as is their expected pay scale.

In sum, there are reasons to be cautious about future employment demand. Present-day programmers come for the most part from a small, well-educated, professional segment of society with rather specialized aptitudes; programming will not provide many opportunities for those who have lost jobs in other sectors. And the very shortage of programmers and their resulting privileged status is giving strong impetus to the development of programming tools that could de-skill programmers or reduce the demand for them.

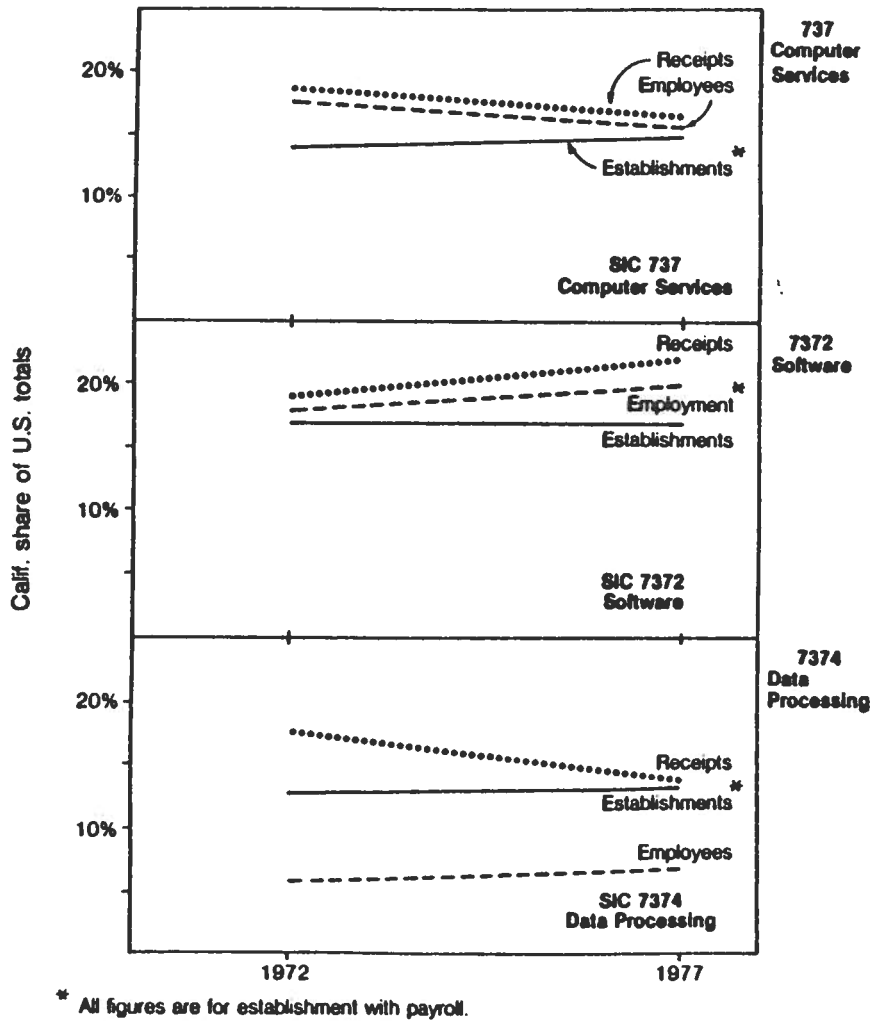
E. Spatial Distribution of Software Jobs

California has about 10% of the U.S. population, but approximately 16% of overall computer services employment and roughly 20% of U.S. employment in independent software firms (SIC 7372) (see Figure 12). The State's share of U.S. total software production in independent software firms rose from 1972 to 1977, while that State's share of data-processing revenues fell and the share of data-processing employment remained the same. As many phases of data processing become more like factory work, other areas of the country with lower costs for real estate, construction, and semi-skilled labor have probably become increasingly attractive.

This interregional pattern is mirrored by similar trends within the state across metropolitan and smaller city locations. The attraction of other parts of the state seems to have affected the Los Angeles area most strongly. As Figure 13 shows, the four-county Los Angeles area's share of California employment in data processing fell off after 1979.

Figure 12

CALIFORNIA SHARE OF U.S. COMPUTER SERVICES ACTIVITY, 1972-77

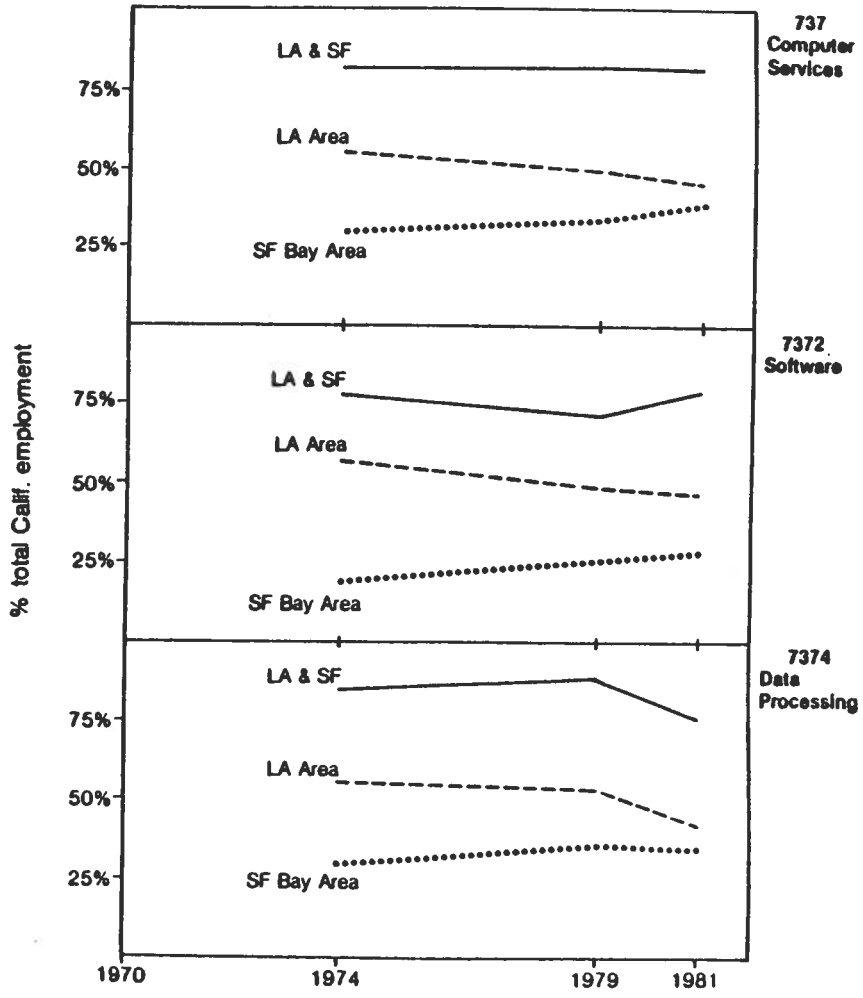


* All figures are for establishment with payroll.

Source Census of Service Industries

Figure 13

COMPUTER SERVICES EMPLOYMENT IN CALIFORNIA: LA AREA, SF AREA EMPLOYMENT; SIC 737, 7372, 7374, 1974-81



Sources County Business Patterns
State of Calif. Emp. Dev. Dept

It also leveled off for the nine-county San Francisco Bay Area. In general there has been, and is still, more software activity in the Los Angeles basin (about 50% of state totals) than in the San Francisco Bay Area (about 35% of State totals), but the Bay Area is gaining in importance. These two metropolitan areas together are the sites for approximately 83% of the software activity in the state.

Measured by employment, the two largest urban centers of the state taken together are losing a share of data processing but gaining slightly in independent software production. Again, this suggests that data-processing activities are going to cheaper locations, whereas software design activities are remaining or concentrating in the urban areas. Software design firms may be showing a preference for urban areas because it is there where their skilled employees prefer to live, and/or where there is a "creative atmosphere" including the presence of major universities. Another factor may be that geographic proximity to large numbers of users is required for firms that must deal directly with their clients while designing or revising and maintaining their software products and services. Evidence on these and related issues from our interviews is included in the next section.

F. Problems and Developmental Barriers

The software sector does face unique development problems. Figure 14 shows the priority ratings of our interviewees on ten basic issues. The five most important ones are briefly addressed in this section.

Figure 14

PRIORITY RANKING OF POLICY AREAS

	Rank	Policy Area	Score *	Comments
PRIMARY	1	Education	50	We are confident that these are primary concerns, and confident of priority rankings.
	2	Copyright	33	
	3	Research & Development	26	
	4	Housing, Transportation, & Infrastructure	24	
	5	Industrial organization & Finance	19	
SECONDARY	6	Telecommunications	12	We are fairly confident that these are secondary concerns, and somewhat confident of priority rankings.
	7	Regulations concerning employee relations	12	
	8	Business Tax Reductions	10	
	9	Software as a capital asset	8	
	10	Software standards	6	

* Values were assigned to interviewee responses as follows:

First Priority = 3
Second or Unranked = 3
Third = 1

1. Education

The images of future programmers provided by our inquiry into the development of new programming tools have important implications for present-day educational policy. There will always be a demand for the kind of specialized education that leads toward Ph.D.s in Computer Science. But the larger middle core of specialists whose job it will be to communicate with computers may have a greater need for basic education than for computer courses. How to think and how to ask the right questions are skills that the computer professional will need to know--not less, but more often than other people. This kind of learning is at least as likely to come out of "the modern version of a liberal arts education," as one source put it, as out of specialized computer-science courses. This recalls to mind the cartoon in which one of a group of chiton-clad youths sitting with their teacher in front of the Parthenon asks, "But, Socrates, will this higher knowledge we seek get us jobs?" As we saw above, the answer is probably "yes."

a. University Education

Universities must respond to both the need for basic education and the need for specialized technical skills. The question is, which specialized skills should be taught, and in what overall balance?

Many of the introductory computer courses at universities are over-enrolled. One University of California introductory computer course has to turn away about 1,000 students per quarter. But the Computer Science Department at UC Berkeley has been hurt by budget cuts and cannot accommodate any more students with its available resources. On

the other hand, a specialized field, such as voice recognition, demands graduate students in linguistics. The University's Language Laboratory, however, has been closed because of cuts in the budget. This means that the development of an area of knowledge that is turning out to be very important to the computer industry has been sharply curtailed.

Thus the simple solution--give Computer Science more money--is a poor solution if the universities' overall budgets are not growing. With an unchanging or shrinking budget, more money to Computer Science will mean less money somewhere else. And if somewhere else is, for example, the Language Laboratory, an area that later turns out to be important to the computer industry, then in the long run everybody loses.

b. Outside the University

One of our interviewees estimated that universities provide computer training for only 20% of the programmers employed in the industry. Several others agreed that it is unrealistic, even undesirable, to expect the universities to supply a greater portion of the need for programmers that they do now. These arguments fit in with the problems, discussed in the previous section, of shifting too many university resources towards computers.

Several alternative ways of training programmers were discussed in our interviews. The most popular idea was that pre-college public education should include more computer training, though the same arguments about computers versus basic education at the university apply to computers versus basic education in primary and secondary schools. Another

proposal was to extend to private industry the apprenticeship program now operating to train computer programmers in State government. Programming apprenticeship in those settings involves two years of on-the-job training and classes and coursework at a community college or university. The program has been highly successful and has reached many individuals who do not fit into the standard molds associated with university education. A third training proposal was to pay industry to train programmers. A fourth was to encourage private training institutes, such as one now operating in Palo Alto, which offers computer courses and is currently negotiating for the right to grant M.S.s and Ph.D.s in computer-related disciplines.

It is clear that in the near term many non-university-trained people will be able to get software jobs if they can get training. The non-university training options described in this section could, for example, facilitate the transition of displaced workers from declining industries to expanding industries. Retraining programs similar to those used in Japan and Germany would greatly help the changeover. In those countries, a worker may receive 80% of his or her former pay for a two-year period of retraining for an occupation in an expanding sector of the economy.

2. Copyrights

To many observers, it has seemed that the problem of product possession is especially acute in the software industry because the product itself is so intangible and could so easily be pirated by employees or users. A contributing problem is that software is protected by

copyright law, not patent law, and that copyright protection is generally weaker than patent protection. Some have felt that weak assurances of product possession have made it harder for software firms to secure financing.

Nineteen of the twenty-two firms we interviewed commented on copyright problems. Twelve said that it was an important problem; seven said that it was not a problem, though some expressed concern that it might become one. Several firms rely on the good faith and self-interest created by employee stock ownership to keep employees from pirating software. One interviewee estimated that 5-10% of software products are now copied, but that this portion would fall as lower software prices reduce the incentive for copying.

3. Research and Development

The problems in R&D are the similar to those in education: how to balance basic and applied research, and how to balance research in fields that are currently of economic importance with research in fields that will become important in the future. For University policy, this problem is related directly to education, since investments in computer education are generally linked to investments in computer R&D. For more general public policy, the balance of short-term and long-term R&D can take the form of a debate about where R&D should take place--in industry or in public institutions such as the University--and about what the relationship between industry and University R&D should be.

Although our interviewees were highly concerned about R&D, their opinions were conflicting. A large number of them seemed to think that

university and other public R&D is important and should have more money, more focus, or both. Most of these also expressed discontent with the current relations between industry and public R&D, citing obstacles to cooperative research projects and poor communication to industry of the results of university and other public R&D.

Others were more interested in promoting private-sector R&D through such means as tax breaks, more government contracts for specific projects, and R&D expenditures designated for small businesses. Some of these seemed to feel that university and other public R&D is not sufficiently productive or efficient.

4. Housing, Transportation, and Infrastructure

With considerable overlap, since most firms gave several reasons, the factors that interviewees said attracted their firms to their present locations are summarized below. Eleven interviewees said that lifestyle is of major importance. Many firms located where their founders had lived for some time before founding the firm. Many of these firms, plus some others, located where the lifestyle choices of prospective good employees converged with the preferences of the firm's founders. Three firms said explicitly, and many implied, that they located near available good employees. Another three mentioned "creative atmosphere" as a factor. For eight firms, being near a university is an important factor for a full range of business and cultural reasons. Two interviewees said that it is not at all important for their firms to be located near a university. Few firms mentioned any need to locate near their market(s).

But such agglomeration benefits may begin to be lost, at least in the Bay Area, as problems with housing, transportation, and overloaded public infrastructure threaten to disperse the industry to other locations, many of them outside the State. This dispersal could hurt the California economy and the overall productivity of the industry. Nine firms said explicitly that housing shortages and/or transportation problems made remaining in the Bay Area increasingly difficult. Some respondents considered extensive telephone networking, in their view, would allow them to locate almost anywhere--so long as the cultural atmosphere was right. A number let some of their employees do some of their work at home, and only two firms expressed a disinterest in encouraging such arrangements. Work at home seems to be more appropriate for smaller programs, e.g., for microcomputers. But many felt that programmers needed at least some group interaction.

5. Industrial Organization and Finance

Some observers have speculated that the software industry could become much more concentrated, with mergers and acquisitions resulting in domination of the market by a few giant firms with huge catalogs of applications packages. Four of our interviewees believed that there was in fact a tendency towards fewer, larger firms in the industry. But twelve interviewees, including two of the four who saw concentration trends, and including people from some of the largest firms, said that there would always be a role for small firms in the industry.

The reasons for the latter belief stem from the organization of the industry. Productivity in small firms is currently higher with current

production processes; smaller software-writing teams have generally proved to be more efficient, and some talented programmers will prefer small firms where there is less hierarchy and less routine. Small firms may be more innovative. They will always be in demand to meet the need for individual software packages. And they will thrive because the capital requirements to develop small- to medium-size programs for small- to medium-size machines are low.

Substantial disagreement arose among industry informants on the scarcity of capital as a developmental barrier. Eight of the twenty-two firms said that they have relied exclusively on internal or family sources for capital. Seven reported using venture capital, and seven used public stock offerings as a major source of capital. In discussions of venture capital, interviewees expressed both strong approval (venture capital brings money, expertise, and connections) and strong disapproval (venture capital demands too large a share of the company, and ignores very small companies.) Most companies complained about banks, or remarked that they could never get bank financing. Some wanted more long-term debt financing, as in Japan.

Many said that the firms had been short of capital during their beginning to intermediate years, when they could have grown faster if they had had easier access to capital. But few wanted state intervention in financial markets. That may be because they had grown so rapidly--in the cases for which we obtained figures, by 15-400% per year. Our firms were probably disproportionately in the "successful" category, and probably on average larger than the small, new firm who might favor improved capital markets.

G. Conclusions and Policy Recommendations

The computer software industry does indeed show promise as a dynamic growth sector with substantial employment prospects. But the structure, behavior, and location of software firms are still in flux. We have been able to identify three critical issues that will determine the industry's future in this country: the existence of an adequate and appropriate labor force; the maintenance of high levels of innovative activity; and the continuation of certain regions' attraction as major software centers. Each has policy implications.

1. Ensuring an Adequate and Appropriate Labor Force

A projection of the size and characteristics of the future software labor force presupposes a vision of the work process, worker productivity, and user features in the high-tech world of the future. In our study we have identified several possible paths for the industry. In reviewing these, we have settled upon what we believe is the most likely development path both for software producers and for users.

The software industry has passed through three stages in the organization of its work process. Once software was unbundled from hardware, the first new software-only firms resembled small craft shops in which one or several skilled professionals worked together to design new products and services. This form still exists and is a source of significant innovation. As some firms matured, particularly on the data-processing side, they began to hire employees for more specific tasks, often more routine and less pioneering, in a form of putting-out system, where programmers could work at home on a piecework basis. In anotner

variation, some large companies, most but not all in the data-processing service subsector, experimented with mixed success at factory-type work processes, where tasks are highly subdivided and programs or services produced on a sort of assembly line. Each of these three types of organization currently exists, yet none will dominate the future of the industry.

The more likely path is one where software products, as opposed to software services, dominate the sales of software firms. These products will become highly sophisticated, with a wide array of variations to fit different user needs. Yet they will also become highly standardized, each marketed to a large number of users, so that firms that invest a great deal in design and innovation can recoup their outlays. Some tailor-made or custom products and services will always be required, but the future will be dominated by these software commodities. In addition, the work on code generators and speech recognition will undoubtedly increase the user-orientation of software. Software labor requirements can thus be expected to shift toward the user side of the market, rather than being registered solely in the software industry per se.

The programming needs of users will be quite different from what they are today. Code generators will maximize the accessibility of computer technology to the professionals who use it--managers in industrial plants, researchers, lawyers, accountants, executives, bankers, financial analysts, etc. Users may employ one or two high level technicians--Ph.D.s perhaps--in the basement, who are available for system intervention. In addition, the user will employ computer specialists to mock up programs and execute test runs. These specialists will

not be required, like today's programmer, to understand computer languages like Pascal, but will be problem solvers and designers, whose expertise includes an understanding of the user's needs. Such specialists may not be expensive, since serious barriers to entry to this occupation, akin to the knowledge of computer languages needed by programmers today, may no longer exist.

What should be stressed in this scenario is the absence of an army of paraprofessional programmers, the retrained auto worker for example, projected by some accounts of high-tech labor demand. In their place will be a bifurcated labor market, with highly skilled Ph.D.s employed in small numbers by software products firms and software users and a large group of user-employed computer specialists whose job responsibilities and pay levels might be akin to those in the clerical workforce today. This scenario has clear implications for the education of future software workers. Rather than retailoring our educational institutions, particularly universities, to meet an immediate demand for programmers with extensive computer language skills, we should be strengthening our basic education programs, especially English, writing, mathematical, and logic skills. We have shown how unsuspected areas of university education, such as languages and linguistics, may be critical to the evolution of software. Our research indicates that all students may benefit from an introductory course in the use of computers, but this knowledge could be mastered in a high school curriculum rather than an introductory college course.

2. Engendering Innovation

Our research indicates that a second issue of central importance to the U.S. software industry's future concerns its ability to remain innovative. In our view, the key to continued innovation is the maintenance of the presence of small, competitive firms in the sector. New firms, often spun off from more established firms, help keep industry leaders on their toes and heighten the innovative spirit. In our research, we were able to document the degree of emerging concentration in the industry better than anyone else to date, and we detected several important problems contributing to the trend toward consolidation.

The software industry taken as a whole does not appear highly concentrated. No one firm claimed more than 5% of the market, and a large number of firms produce software products and services. Nevertheless, two factors caution against celebrating the competitiveness of the industry. First, we found through our interviews that hundreds of market niches actually exist in the industry, particularly in the applications software arena. In many of these, the market may be met entirely by one or very few firms. Indeed, our informants suggested that they deliberately seek out a market niche in order to ensure returns on their investment of time and energy. While at the current time many market niches remain unsaturated, the prospects are that some degree of monopoly power may accrue in the future to those firms dominating each niche.

Secondly, a trend toward consolidation can be detected. Often this takes the form of mergers or buyouts, but it is also the result of large hardware, communications, and electronics firms setting up their own

software divisions. While the number of firms entering the industry is on the rise, the percentages of output and employment accounted for by large multi-establishment firms are increasing. Yet our research suggests that such concentration is not warranted by the purely production issues in the industry, but by several problems faced by the small, innovative firm.

First, the present copyright laws offer firms little protection for their new products. Software is treated as written material, not a commodity, so that the more lenient copyright laws apply, rather than patent restrictions. Small firms face stiff legal costs if they pursue litigation. Many feel compelled to expand forward into sales and distribution to recoup their investment; some are driven to join a larger firm with a legal department and its in-place distribution system.

Second, firms in the initial stages of development face great difficulties raising capital. Our sample was drawn almost entirely from more mature firms that had surmounted these early development costs, yet access to investment funds clearly remains a problem for many would-be innovators. A firm at the startup point has no assets to speak of, merely a team of eager and innovative personnel, which is not usable collateral for a bank loan. Even when programs exist in written form, banks refuse to consider them assets, again precluding access to bank funds. Capital needs at the beginning tend to drive small firms into mergers with larger firms that have internal funds at their disposal.

Third, formidable marketing problems confront small firms. A good product is not a money-maker unless it reaches its market. Most small firms have neither the expertise nor the resources to reach their

targets. Many are thus driven toward merging with other companies who possess good distribution links and have access to broader geographical areas. Large firms, too, are driven by the desire to offer a full spectrum of products encouraging buyouts of smaller, more youthful firms. Computer hardware companies, in particular, are drawn to higher profitability software and continue to assemble hardware/software packages for users.

One important countervailing force favoring the small, innovative firm was detected in our study. Programmers as a group are a highly motivated, entrepreneurially minded group. Many have been drawn to the occupation precisely because of the chance to work in small groups and to have substantial control over their hours, product, and work environment. Despite attempts by many firms to enhance worker satisfaction through profit sharing and perquisites, turnover is high. High turnover is especially damaging to a software producer, and turnover rates appear to be greatest in larger firms and those with more of a factory-like environment.

The upshot of these trends is that small firms, although the source of much innovation, continue to struggle for their existence. Already, substantial concentration is emerging in systems software, while new entrepreneurship is most evident in the single-task applications package for the smallest computers. The trend toward software products, reviewed above, will intensify competition in this latter sector. Some niches for custom-design of software for industrial users will undoubtedly continue to exist, similar to today's machine tool industry. But for most users, especially for institutional purposes such as banking

and insurance, packaged software will predominate. The forces encouraging concentration and discouraging innovation raise several policy issues, returned to in the final section below.

3. Maximizing the Benefits for National and Regional Economies

A separate issue is the degree to which we can expect the software industry to make substantial contributions to the U.S. and individual state economies. An industry may do so by providing job growth, by increasing the tax base through both income and profit generation, and through forward and backward linkages with other supplying and purchasing sectors. In the case of software, direct job creation is the major benefit foreseen, since the industry makes few physical plant additions to the tax base and has few backward linkages to suppliers, except to the extent it depends upon the computer industry. But software may be able to hold and draw software users to its innovative centers. Yet of all high-tech sectors, software is the most labor-intensive and the most promising on the employment front.

A central issue for high-tech states is whether or not projected job growth in the software industry will occur within their boundaries. A related issue is where, at both state and federal levels, policymakers should try to direct new growth. Competition from both England and Japan could result in software imports replacing domestic production. In addition, firms confronted with high costs or marketing disadvantages could decide to decentralize employment to other regions. It is our conclusion that existing centers currently possesses substantial locational advantages, and that changes in software products and in

communications technology will probably reinforce that edge. Nevertheless, the degree to which any one state maintains its software preeminence will depend upon the availability of an adequate labor force and the preservation of its innovative performance.

Our research confirms the general pattern believed to characterize all high-tech, innovative industries: firms are most likely to cluster together in a few locations. These spots generally have a top-rate university nearby (in our interview cases, these were Berkeley and Stanford), a substantial technical/professional labor force, good access to national communications and transportation systems, and urban/environmental amenities likely to draw and hold a professional labor force. In addition, we were able to isolate certain specific features of software industry structure and needs that reinforce or counteract this tendency.

First, we detected a decentralizing force in the recent emphasis on marketing and servicing activities. Currently, a firm's need to reach national markets and to educate and consult with users often results in the dispersal of new jobs to regional centers nearer to users. This makes sense, as long as these activities remain a major preoccupation. Our research suggests, however, that they will not be as important in the long run, especially with the advent of the user-friendly technologies.

A second decentralizing force is the pre-existing concentration of a client in a particular location. Software firms serving the airlines, for instance, have followed the airline company headquarters to Texas. The migration of software in search of centralized clients is most

common on the service side of the industry and not as characteristics of software products. Again, the shift toward software products will tend to reinforce initial locational advantages, if the industry remains innovative in these centers.

A third force for decentralization is the widening differential between cost of living in high-tech metropolitan areas and other locations. Particularly problematic is the high cost of housing for professionals in those locations where software activity is clustered. Our informants echoed the concerns of the semiconductor and computer industry on this score, especially with respect to Silicon Valley. Yet other responses in our interviews suggested that professionals were quite willing to pay a premium to live in California, and particularly in urban areas, both because of nearby job alternatives and because of the general cultural and recreational advantages of the region. We believe that most firms are constrained by their employees' desire to remain in the Los Angeles area or the San Francisco Bay Area, despite the attraction of lower costs incurred at alternative sites.

In addition to the factors just mentioned, changes in telecommunications will also reinforce the pull of original centers. The continued improvement in telecommunications technology detaches software producers from clients spatially. Firms informed us that they expected to rely more and more on phone diagnosis of software problems rather than face-to-face service visits. This can be done as cheaply by long-distance telephone as by an intraregional call. The net effect is to reinforce large-city communications centers. We originally expected that the same communications technology would detach programmers physically from

software firms, but we encountered great skepticism in this regard from our interviewees. Most believe that programmers will be required to show up in the central office several times a week, to ensure communication and work progress.

One additional center-reinforcing factor emerged from our interviews. Young software firms are concerned about their "address." They want to be identified with the heart of software inventiveness. For this reason, they prefer a Palo Alto or San Francisco address. For the same reason, unfortunately, most would shun the older city centers like Oakland, precisely those sites where unemployment is highest and jobs most in demand. Only for software firms involved with large-scale data processing, such as software for the insurance industry, are sites in places like Emeryville and Oakland attractive, because rents and secretarial costs are lower.

4. Priorities for State Policy

Three different perspectives on state policy are possible. The first encompasses the views, where consensus exists, of the industry itself. These views (and the controversial ones as well) are reported above in the chapters summarizing our interviews. In brief, firms currently populating the industry are concerned above all with the availability of well-educated programming talent. In addition, they favor tightening copyright laws, increasing basic university research, lowering the cost of living for their employees, and amplifying the financial resources available to growing firms.

A second perspective views the strength and job-creation potential of the industry as important state economic development goals, but takes a longer-run view of the industry's future, which may differ from the immediate expressed concerns of today's competitors. We believe that there are two issues that distinguish this posture from the one which emerged from our interviews. First, the spectre of tremendous unmet demand for programmers contradicts our informants' views of future work structure in the industry. The programmer as we know him/her today may be relatively unimportant in the occupational structure of the future. Therefore, this perspective cautions against retooling educational institutions to rapidly increase the supply of programmers.

Second, our sample of successful firms tended to dismiss the need for state intervention either into the capital market or into the marketing process. Many believed that government should leave the industry alone. Yet other, smaller and less successful firms may be unable to compete, and wistful would-be entrepreneurs unable to set up shop, because they confront the front-end capital problem or cannot break into an established distribution network. A healthy, competitive, innovative industry may require some form of governmental support for start-up capital and for joint marketing arrangements that would counteract the tendency towards concentration that we have documented.

A final perspective places the software industry's promise and problems in the context of the larger economy. No industry should be treated as a policy target without considering the claims of other industries, the impact of that industry on others, and the general costs and benefits it will bring to the society at large. Several issues of

this sort are attached to the software industry. First, the software industry is a key component in the rapid mechanization of many other industries. Robotization requires sophisticated programs to direct mechanical arms to do what human labor has done for hundreds of years. While mechanization may eliminate difficult and dangerous work, and increase labor productivity, it also displaces workers who often have a difficult time finding alternative employment. We have found that very few industrial workers are apt to secure jobs in a new sector like software. Therefore, it is worth asking whether accelerating the rate of software innovation might not exacerbate adjustment problems in other sectors. This question lay outside the purview of our research, but we believe it is an important one for policy makers.

A second issue from this perspective is the effect on workers within the industry. Time constraints precluded a set of interviews with software programmers. We believe that there are a number of issues concerning health and safety (eye strain, stress, etc.) and job satisfaction that are critical to workers in the industry. Research and policy efforts should address these concerns. It is possible, too, that policies that accelerate the development of user-friendly software will eliminate the programmer's role, deskilling or displacing him in the process. State policy should avoid repeating the mistakes of a century of agricultural research that was largely dedicated to increasing crop productivity through mechanization, hybrids and pesticides, without considering the impact on agricultural workers.

State policy should also avoid enhancing business prospects for software firms at the expense of their customers. An example where this

could occur concerns the copyright area. While innovative firms need protection of their investments, stringent copyright provisions can also lead to monopoly profits and higher prices, which could impede the contribution software makes to productivity in other sectors. Similarly, state policy should avoid responding to complaints about the high cost of housing and transportation for this group of relatively highly-paid professionals without also considering the more pressing housing and transportation needs of lower-income groups in the population.

The limits of our research focus do not permit us to pass judgment on the tradeoffs between supporting a high-tech sector such software and other demands on the state budget. We believe that software will be a net job creator, at least for another decade, but we do not believe that it will be a panacea for industrial restructuring problems in the California economy. The best way for the state to aid the industry, we repeat, is to strengthen its commitment to basic education and employing selective policy tools to engender innovation and competition.

Figure A-1

SOFTWARE FIRMS INTERVIEWED

PRIMARILY SOFTWARE

Name of Firm	No. of Emp.	Computer Size			Software Type		Location of Corporate Headquarters	Age of Firm
		main frame	mini	micro	op. sys.	appl. pkg.		
Bode & Babbage	150	X			X		Sunnyvale, Calif.	15 yrs.
Software Module Marketing	60	X			X		Sacramento, Calif.	8 yrs.
Software Pursuits	18	X			X		San Francisco, Calif.	7 yrs.
Argonaut	15		X	X		X	Oakland, Calif.	11 yrs.
Ask	105		X			X	Los Altos, Calif.	10 yrs.
Comserv	325	X	X			X	Minnesota	12 yrs.
Insumet	300		X			X	Emeryville, Calif.	9 yrs.
Relational Technology	20		X			X	Berkeley, Calif.	2 yrs.
Digital Research	150			X	X	X	Pacific Grove, Calif.	6 yrs.
Information Unlimited Software	25			X		X	Marin County, Calif.	4 yrs.
Micropro	350			X		X	San Rafael, Calif.	4 yrs.
Visi Corp	130			X		X	San Jose, Calif.	4 yrs.

Figure A-1 (continued)

HARDWARE & SOFTWARE

Name of Firm	No. of Emp.	Computer Size			Location of Corporate Headquarters	Age of Firm
		main frame	mini	micro		
IBM	330,000	X	X	X	Armonk, NY	~70 yrs.
Hewlett-Packard	64,000		X	X	Palo Alto, Calif.	~45 yrs.
ROLM	4,400		X		Santa Clara, Calif.	14 yrs.
Atari	~5,000			X	Sunnyvale, Calif.	10 yrs.
Apple	2,500			X	Cupertino, Calif.	7 yrs.
Intel	17,000			X	Santa Clara, Calif.	14 yrs.

SOFTWARE & PROFESSIONAL SERVICES

Name of Firm	No. of Emp.	Services & Products	Location of Corporate Headquarters	Age of Firm
Berkeley Solar Group	20	Custom software; consulting on energy use in architecture	Berkeley, Calif.	8 yrs.
Berkeley Systems Works	20	Custom software; research; consulting	Berkeley, Calif.	4 yrs.
Inno Sys	7	Custom software; research; consulting	Berkeley, Calif.	9 yrs.
Structural Software Development	11	Structural engineering analysis and software	Berkeley, Calif.	4 yrs.

BIBLIOGRAPHY

BOOKS

Boehm, B. W. 1981. Software Engineering Economics. New Jersey: Prentice-Hall.

Fishman, K. D. 1981. The Computer Establishment. New York: Harper and Row.

Korites, B. J. 1982. How to Sell Your Micro Software. Massachusetts: Kern Publications, 1982.

Stern, R. A., and N. Stern. 1981. An Introduction to Computers and Information Processing. New York: John Wiley and Sons.

PERIODICALS

Business Week. July 1, 1980.

Business Week. "Missing Computer Software." September 1, 1980.

Business Week. "The Risky Business of Selling Homegrown Programs." March 9, 1981.

Business Week. "Briefs: The U.S. Market (Information Processing)". July 5, 1982.

Dataguide. Spring 1981.

Datamation. August 1981.

Datamation. "The Datamation Top 100." June 1982.

The Economist. "Tortoises and Hares in the Personal Computer Race." February 20, 1982.

Electronics. "Computer Technology Shifts Emphasis to Software: A Special Report." May 8, 1980.

Electronics Review. September 8, 1981.

Financial World. "Is the Computer Software Boom For Real?" November 15, 1981.

Forbes. "Tomorrow Has Arrived." February 15, 1982.

Mini-Micro Systems. "Hardware Vendor Offers Equity for Software." May 1981.

Mini-Micro Systems. "Hardware Vendors Scramble for Software." January 1982.

NEWSPAPERS

New York Times. January 8, 1981.

New York Times. "Computers: The Action's in Software." November 8, 1981.

San Francisco Chronicle. "The Boom in Computer Software." October 18, 1981.

Software News. "Software Boom Seen Continuing to Mid-80s." December 7, 1981.

GOVERNMENT DOCUMENTS

Center for the Continuing Study of the California Economy. March 1982. California's Technological Future: Emerging Economic Opportunities in the 1980s. Report to the California Department of Economic and Business Development.

SRI International. March 1982. California's Technological Future: Emerging Economic Opportunities in the 1980s. High Technology and the California Workforce in the 1980s. Submitted to California Department of Economic and Business Development.

State of California, Employment Development Department, Employment Data and Research Division. Occupational Employment in Manufacturing Industries. Occupational Employment in Nonmanufacturing Industries (Supplement).

State of California, Employment Development Department. Unpublished data analyzed by authors in May 1982.

State of California. Preparing Students for California's Changing Economy. April 1982.

U.S. Department of Commerce, Bureau of the Census. 1977 Census of the Service Industries.

U.S. Department of Commerce, Bureau of the Census. 1972 Census of the

Service Industries.

U.S. Department of Commerce, Bureau of the Census. County Business Patterns, California. 1974, 1977, 1979.

U.S. Department of Labor, Bureau of Labor Statistics. July 1979. Employment and Earnings, United States, 1909-78. Bull. 1312-11.

U.S. Department of Labor, Bureau of Labor Statistics. Employment and Earnings. March 1980, 1981, 1982.

SPECIAL REPORTS

Association of Data Processing Service Organizations (ADAPSO). Sixteenth Annual Survey of the Computer Services Industry. August 1982.