Chapter 3

The Federal Government Role in the Development of the American Software Industry: An Assessment[1]

Richard N. Langlois and David C. Mowery

1. Introduction.

The development of the U.S. computer software industry has been powerfully influenced by federal government policy during the postwar period. Its importance for the demands of Cold War defense, especially strategic air defense during the 1950s, meant that the software industry received considerable support from federal R&D and procurement funding throughout the postwar period. But the very novelty of computer technology and software meant that a substantial portion of the defense-related spending in software was allocated to the creation of an infrastructure for the support of a new area of R&D, training, and technology development. Federal support for the creation of this infrastructure provided important benefits to the commercial U.S. software industry. From the earliest years of the postwar era, private industry has been responsible for a great deal of innovation in software; but by the 1960s, these industrial innovations drew on research and manpower that had been generously supported by federal government funds.

Because of the complex and changing relationship between software and hardware technology during this period, much of the influence of federal government policies on the software industry was channeled through programs affecting the overall computer

industry. For example, federal agencies' development and procurement expenditures for hardware included spending on software for much of the postwar period.

The structure and effects of federal policy toward the U.S. software industry have a number of similarities with the history of federal support in other postwar U.S. high-technology industries. As in the cases of airframes (Mowery and Rosenberg 1982), semiconductors (Levin 1982), and computer hardware (Katz and Phillips 1982; Flamm 1987), defense-related support for both R&D and procurement accelerated the early development of the software industry. As in those other cases, the influence of defense-related procurement on the commercial software industry declined as defense needs diverged from those of a burgeoning commercial market. Indeed, throughout the 1980s, Pentagon policymakers sought ways to tap commercial software applications and operating systems for military systems. There are also some important differences between the software and other high-technology industries, however, most notably in the nature and evolution of military-civilian spillovers within software, as well as in the role of federally funded university research within the industry's development.

Rosenberg (1992) has argued that the computer is one of the most significant examples of a large class of scientific instruments that have been developed in universities and widely applied in industrial economies. Rosenberg's observations concerning the "instrumental" nature of the computer are borne

out by the history of federal policy in software innovation in at least two ways: (1) the key role of the university within the software industry; and (2) the importance of federal support for universities' adoption of the mainframe computer, the critical instrument for software research and innovation.

In contrast to their stereotypical role as performers of basic research, university researchers pioneered in the development of computer technology well in advance of industry in both the U.S. and Great Britain. British and U.S. universities and research institutes affiliated with them were responsible for important advances during the late 1940s and early 1950s in computer architecture and hardware, including the stored-program concepts that were the origins of software. Much of the federal government's early postwar efforts to develop computer technology relied on university researchers.

In both nations, but especially in the U.S., technological advances and researchers from universities entered the domestic electronics industry, and industry came to dominate the development of subsequent generations of hardware. Universities remained important, however, in many software advances from the mid-1950s onward. The contributions of U.S. universities to these developments relied on the growth of a new academic discipline, computer science. The creation of this academic field was aided by federal support during the 1950s and 1960s for the purchase of the scientific instrument that was indispensable to computer science research, the mainframe computer.

The software industry, like other postwar high-technology U.S. industries, drew on defense-related support for applications development and basic research. Direct "spillovers," i.e., widely adopted civilian versions of software developed initially for military applications, did appear, but these were supplemented (Flamm, 1988, p. 26) by a number of advances from the private sector. A number of other advances (including several important programming languages and operating systems) were developed in universities with federal funding.

Growing concern within the Department of Defense over the soaring costs, project delays, and unreliability associated with complex, software-intensive weapons systems (concerns that were heightened by the Strategic Defense Initiative of the 1980s) led to two DoD initiatives during the late 1970s and 1980s--a program in "software engineering," and (some years before similar efforts in other "dual-use" technologies) efforts to utilize commercial software for military applications. The limited success of the second of these initiatives thus far suggests that similar objectives in such technologies as semiconductors or flat-panel displays will prove difficult to attain.

Throughout the postwar period, the federal government has accounted for a large share of total U.S. demand for software. Flamm (1987, pp. 122-123) estimated that the federal government was the largest single U.S. customer for traded software in 1982. More recent data on market trends are not available, but it is likely that the federal government's share of the U.S. market for

traded software has declined during the past decade. A great deal of defense-related software procurement has involved the purchase of "embedded" software. There are relatively few examples of major "standard" operating systems, programming languages, or applications being developed initially for federal agencies. But the development of custom software and services for federal purchasers was for much of the 1960s and 1970s a rapidly growing industry in the Washington D.C. region.

Our discussion of the federal role in the U.S. software industry focuses primarily on policies that directly affected this industry; for this reason, and because it is covered in another chapter of this volume, we omit consideration of intellectual property protection. We begin with a description of the earliest years of federal involvement in the U.S. computer industry, a period during which software scarcely existed as a distinguishable technology and focus of development effort. We then examine the federal role in supporting the emergence of computer science as an academic discipline, a discussion that spans 1955-90. The role of the Department of Defense during this period is the subject of the next section. The penultimate section briefly discusses the activities of another important U.S. agency in software development and procurement, the National Aeronautics and Space Administration (NASA). The concluding section summarizes our argument and suggests some points of contrast with the experience of other industrial economies.

2. The early years.

Software technology did not develop in a political or institutional vacuum; nor was government policy unaffected by changes in the nature of software technology. Despite a number of prewar precursors, the history of computer hardware and (eventually) software development begins with World War II. During the war years, the American military sponsored a number of projects to develop computers to solve special military problems. The ENIAC--generally considered the first fully electronic digital computer--was funded by Army Ordnance, which was concerned with the computation of firing tables for guns. MIT's Whirlwind computer, which was introduced in 1951, grew (after a difficult adolescence) out of a project begun in 1943 to create an analog-computer flight simulator for pilot training.

In the earliest days of postwar computer technology, software was literally indistinguishable from hardware. Software was effectively born with the advent of the von Neumann architecture for computers. In the summer of 1944, the mathematician John von Neumann learned by accident of the Army's ENIAC project. Developed by J. Presper Eckert and John W. Mauchly at the Moore School of the University of Pennsylvania, the ENIAC did not rely on software, but was hard-wired to solve a particular set of problems. Von Neumann began advising the Eckert-Mauchly team, which was working on the development of a new machine, the EDVAC.

Out of this collaboration came the concept of the stored-program computer: instead of being hard-wired, the EDVAC's instructions were to be stored in memory, facilitating their modification. As we would now say, the computer could be programmed by software rather than hardware. Von Neumann's abstract discussion of the concept (von Neumann, 1945) circulated widely and served as the logical basis for virtually all subsequent computers. But even after the von Neumann scheme became dominant, which occurred rapidly in the 1950s, software remained closely bound to hardware. During the early 1950s, the organization designing the hardware generally designed the software as well. As computer technology developed and the market for its applications expanded after 1970, however, users, independent developers and computer service firms began to play prominent roles in software development (see Chapter 2).

Although military support for the ENIAC and Whirlwind projects began with narrowly defined goals, these programs produced general principles and technologies that found much broader application. Indeed, in the case of Whirlwind (Redmond and Smith 1980), the Navy never obtained its hoped-for flight simulator. Jay Forrester, who took charge of the project in 1943, became embroiled in a prolonged struggle with the Office of Naval Research (the primary postwar Navy research funding agency) over his desire to shift the project to the development of a general-purpose digital computer rather than a flight simulator. The Whirlwind project, which was by far the most expensive of

the early postwar federal computer programs,[2] was spared only when the U.S. Air Force adopted it as the basis for the SAGE air-defense program that began in the early 1950s. In addition to driving the development of a reliable large computing system and the communications technologies necessary to link these computers with radar networks, SAGE was among the earliest programs in large-scale software development (Tropp, 1983).

The development of a U.S. software industry really began only when the first stored-program computers begin appearing in significant numbers. The first fully operational stored-program computer in the U. S. was the SEAC, a machine built on a shoestring by the National Bureau of Standards in 1950 (Flamm, 1988, p. 74). A number of important machines followed. Among these, in addition to Whirlwind and ENIAC, were:

* The IAS computer, 1951, built by von Neumann at the Institute for Advanced Study and "cloned" at the RAND Corporation and four national labs. Funding came from the Army, the Navy, and RCA, among others.

* UNIVAC, 1953, built by Remington Rand, which had bought the rights to the Eckert-Mauchly technology. Early customers included the Census Bureau and other government agencies as well as private firms.

* The IBM 701, 1953, developed by IBM and influenced by the IAS design. Originally developed as a scientific computer for the Defense Department, who bought most of the first units.

The most commercially successful machine of the decade, with sales of 1800 units, was the low-priced IBM 650 (Fisher et al. 1983, p. 17). The 650, often called the Model T of computing, thrust IBM into industry leadership (Katz and Phillips 1982, p. 178; Flamm 1988, p. 83). Even in the case of the 650, however,

government procurement was crucial: the projected sale of 50 machines to the federal government (a substantial portion of the total forecast sales of 250 machines) influenced IBM's decision to initiate the project. The large commercial market for computers that was created by the 650 provided strong incentives for industry to develop software for this architecture.

Programming all of these early machines was a tedious process that resembled programming a mechanical calculator: the programmer had to explicitly specify in hardware terms (the memory addresses) the sequence of steps the computer would undertake. This characteristic tied program development closely to a particular machine, since programmers had to understand its hardware architecture. Since few models of any single machine were available, programming techniques developed for one machine had very limited applicability. This was one reason why the commercial success of the IBM 650 was crucial to advances in software and in programming techniques. As Goldstine (1972, p. 331) points out, the 650 created a generic "platform" for the development of programs that could run on a large installed base.[3]

Perhaps the main bottleneck of this "machine-language programming," however, was the difficulty of changing a program. Inserting new data or instructions into the sequence required changing most if not all memory references. In response to this problem, programming tools--rudimentary languages--appeared. For example, researchers at IBM and in the Whirlwind group at MIT

developed symbolic assembly languages in which coded statements referred to "symbolic" addresses that the computer converted to specific hardware locations (Sammet 1969, p. 3; Bashe, et al. 1986, pp. 323-338).

In general, the direction of technological advance was toward "higher-level" languages, that is, languages employing a relatively user-friendly notation that software later translated into machine language. These included assemblers, interpreters, and compilers. The last is a program that translates a higher-level "source" code into a machine-language "object" code that a specific computer can understand. In this way a higher-level language can become machine-independent, since different compilers can translate the same source code into different machine languages. The first commercial compiler was the A-0, developed for the Remington Rand UNIVAC (Bashe, et al. 1986, p. 340) by Grace Murray Hopper, who described it as a means to support "automatic programming," using the computer to automate some of the tasks of machine-language coding (Hopper 1954, cited in Bashe, et al. 1968, p. 431; see also Sammet 1969, p. 13 and Hopper 1981, passim).

Much of the early work on automatic programming was conducted not at universities but at the laboratories of computer makers or users. Hopper, a veteran of the wartime military Mark I project at Harvard, was supported by Remington Rand in developing the A-0 compiler. The assembler most widely used on IBM machines was developed by a user (United Aircraft) and

disseminated through an IBM users' group called SHARE (Bashe et al. 1986, p. 358). FORTRAN, the first genuine higher-level language, also was developed by IBM researchers (Backus 1981). And the earliest IBM operating system--the program that stage-manages the execution of programs and the use of peripherals--was written at the General Motors Research Laboratories (Bashe et al. 1986, p. 359). An important exception to this general pattern was MIT, where in the early 1950s the Whirlwind group developed not only a symbolic assembler but also an operating system and an algebraic compiler that anticipated some of the capabilities of FORTRAN (Flamm 1988, table A4; Backus 1981). But its unique architecture and operation solely as a research instrument meant that, unlike those for commercial machines, the automatic-programming techniques developed for Whirlwind had significantly less influence than their private-sector counterparts.

The federal government influenced the development of early automatic programming techniques through its support for the dissemination of information. From the earliest days of their support for the development of computer technology, the U.S. armed forces were surprisingly anxious that technical information on many aspects of this innovation reach the widest possible audience, in some contrast to the military in Great Britain or the Soviet Union.[4] The Office of Naval Research organized seminars on automatic programming in 1951, 1954 and 1956 (Rees 1982, p. 120). Along with similar conferences sponsored by

computer firms, universities, and the meetings of the fledgling Association for Computing Machinery (ACM), the ONR conferences circulated ideas within a developing community of practitioners who did not yet have journals or other formal channels of communication (Hopper 1981). The ONR also established an Institute for Numerical Analysis at UCLA (Rees 1982, p. 110-111), which made important contributions to the overall field of computer science.

The private sector took some of the first steps to begin building the discipline of computer science within U.S. universities. Computer manufacturers recognized that in addition to the public-relations benefits of supporting higher education, they could increase demand for their products by facilitating the acquisition and use of their hardware at universities (Fisher et al. 1983, p. 169). Support of academic computing would attack the software bottleneck through training more programmers and "lock in" future users and buyers of computer equipment.[5]

For example, in addition to offering price discounts on its machines, Control Data Corporation (CDC) offered research grants, free computer time, and cash contributions to U.S. universities (Fisher et al. 1983, p. 170). In addition to donating computer time to establish regional computing centers at MIT and UCLA in the mid-1950s,[6] IBM rented some 50 of its model 650 computers to universities at reduced rates[7] (Galler 1986; Fisher et al. 1983, pp. 170-172). For example, the IBM 650 at the Carnegie Institute of Technology's new Graduate School of Industrial Administration

that was used by Herbert Simon, Allen Newell, and Alan Perlis in their early work on artificial intelligence was acquired with funds from private foundations, although Simon and others also received support as consultants to the Rand Corporation (Bach 1986).

The institution-building efforts of the National Science Foundation and the Defense Department came to overshadow private-sector contributions by the late 1950s. In 1963, about half of the $97 million spent by universities on computer equipment came from the federal government, while the universities themselves paid for 34 per cent and computer makers picked up the remaining 16 per cent (Fisher et al. 1983, p. 169).

The federal government's expanding role in supporting R&D, much of which was located in U.S. universities, during the 1950s, was supplemented by procurement spending on military systems. In this area, the government's needs differed from those of the commercial sector, with important implications for the "production technologies" employed in defense and commercial software. Defense-related demand for software (outside of logistics applications) was aimed initially not at general-purpose automatic programming tools but at special-purpose, large-scale software for specific defense missions, as in air defense.

The most conspicuous early example of defense-related software development and procurement is the SAGE air-defense system, the computerized early-warning system developed and

deployed in the 1950s, which involved what was by far the largest programming effort of the day. In 1950, the Air Force established the MIT Lincoln Laboratories to develop air-defense technology. This effort absorbed MIT's Whirlwind project and evolved into SAGE, the Semi-Automatic Ground Environment. Although the Whirlwind had long since severed its connection to the flight-simulator project, it had been designed for real-time command-and-control applications rather than for batch processing, and was one of the first examples of a "mission-critical" defense computer. Successful tests of the SAGE system on Cape Cod led to a full-scale development effort in 1953, coordinated by Lincoln Labs. Lincoln Labs chose IBM to produce operational computers that were based on the Whirlwind model; AT&T developed the communications system that linked the radar units; and Burroughs built peripheral equipment. A division of the Rand Corporation that soon spun off to become System Development Corporation (SDC) took up the massive programming task.[8]

The RAND group that became SDC started out as a psychological-testing unit engaged in simulating human-machine interaction in radar-defense installations. In addition to its simulation experience, RAND in 1955 already employed what one official estimated to be 10 per cent of all the qualified programmers in the country--about 25 people (Baum 1981, p. 23). By 1959, SDC had more than 800 programmers working on SAGE (Baum, 1981, p. 35). By 1963, SDC had 4,300 employees (not all of whom

were programmers) and, more significantly, some 6,000 former employees spread throughout the computer industry (Baum 1981, p. 47). One of the greatest contributions of SAGE was its training of a large cadre of educated systems programmers. Indeed, because SDC was restricted by Air Force pay scales and because it sought to play this training role, the company encouraged turnover, which ran to 20% per year. (Baum 1981, p. 51). As one SAGE veteran noted in the early 1980s, "the chances are reasonably high that on a large data processing job in the 1970s you would find at least one person who had worked with the SAGE system" (Bennington, 1983, p. 351).

SAGE also contributed to the embryonic discipline of software engineering. Although many claim that this discipline was born at a 1968 NATO conference (Naur, Randell, and Buxton, 1976), SDC developed many of the programming and organizational techniques later associated with software engineering. These included modular design techniques that facilitated task decomposition and organized the division of labor in large projects.

SAGE was the first of many large-scale government programming projects. After SAGE, SDC undertook the development of a command-and-control system for the Strategic Air Command that required a then-astounding one million lines of code. SDC also developed JOVIAL, a higher-level programming language for command-and-control applications that was widely used in industry. By 1960, however, SDC began to face competition from

the vertically integrated software divisions of large firms like Boeing and TRW and from the more than 2000 firms that had begun to enter the contract-software business (Cusumano 1986, p. 121). SDC abandoned its nonprofit status in 1969 and eventually merged with Burroughs (later part of Unisys) in 1981.

The federal government remained a major purchaser of contract software well into the 1980s. By one estimate, DoD spent some $4 to $8 billion on contract software in 1982 (Flamm 1987, p. 123).

3. Creating an Academic Discipline: Computer Science and the Software Industry

The other chapters in this volume emphasize the role of universities in the growth of the software industries of Western Europe, Japan, and Russia. Universities have been important sites for applied, as well as basic, research in software, and have contributed to the development of new hardware. In addition, of course, the training by universities of engineers and scientists active in the software industry has been extremely important. By virtue of their relatively "open" research and operating environment that emphasizes publication, relatively high levels of turnover among research staff, and the production of graduates who seek employment elsewhere, universities can serve as sites for the dissemination and diffusion of innovations throughout the global software industry.

U.S. universities provided important channels for cross-fertilization and information exchange between industry and

academia, but also between defense and civilian research efforts in software and in computer science generally. Hendry (1992) argues that a lack of interchange between military and civilian researchers and engineers weakened the early postwar British computer industry;[9] the very different situation in the U.S. enhanced the competitiveness of this nation's hardware and software industry complex. The more modest role of universities in computer science and software-related research activities in Japan and the Soviet Union also reduced somewhat the flow of knowledge among different research sites and hampered the pace of technological progress in these nations' software industries (See Chapters 5, 6 and 9).

Federal policy contributed to the central role of U.S. research universities in the advance of hardware and software technologies. As our previous discussion of the earliest years of the U.S. computer industry suggests, universities were among the first developers of computers, supported by wartime and early postwar R&D contracts from the federal government. But even after the rise of a substantial private sector industry dedicated to the development and manufacture of computer hardware, federal R&D support aided the creation of the new academic discipline of computer science. The creation and legitimation of a new academic discipline, particularly in the applied and engineering sciences, within U.S. universities is itself hardly novel. Partly because of their decentralized structure and financing, U.S. universities frequently have responded to the demands of

industry (and, in some cases, the state governments that supported so many U.S. universities) by developing new academic departments and disciplines in areas such as chemical engineering, electrical engineering, and aeronautical engineering (Rosenberg and Nelson, 1994; Mowery and Rosenberg, 1993). Private firms supported the early development of academic computer science, but their contributions and support soon were outweighed by those of the federal government. Much of this government financial support was motivated by defense concerns.

Comprehensive data on federal R&D support for academic research in computer science, let alone in software per se, are difficult to obtain. The data in Figure 1 display trends in total federal support for computer science R&D during fiscal 1959–1971, and reveal the importance of DoD-related sources of funding throughout this period. Figure 1 points out the rapid rise of the Advanced Research Projects Agency (ARPA, long known as DARPA), established in 1958 to conduct long-range R&D of interest to all of the uniformed services, in supporting computer science R&D. Since its data include federally supported R&D performed outside of universities, the Figure understates the importance of NSF as a funder of academic computer science research. Nevertheless, according to Yudken and Simons (1988), defense-related agencies accounted for more than 50% of academic computer science R&D from fiscal 1977 through the mid-1980s, and defense-related support for applied computer science research grew rapidly after fiscal 1983.

Figure 1 about here.

The foundations for the contributions of U.S. universities to the growth of the software industry were laid during the 1950s by two federal agencies: NSF and ARPA. The approaches taken by these agencies to building a new academic discipline complemented one another, as NSF's support was distributed broadly and that of ARPA was concentrated on a few leading research universities.

As Aspray and Williams (1993a) note in their survey of the early NSF programs in computer science, during the early 1950s, NSF support for computer science was modest and was channelled through its mathematics research program. This picture changed as a result of the 1956 endorsement by the Advisory Panel on University Computing Facilities (chaired by John von Neumann) of a specialized NSF program for the support of computer science, the 1957 launch of Sputnik, and the passage of the National Defense Education Act in 1958. NSF support for computer science research grew rapidly after 1958, and was especially important in meeting the critical need of academic researchers for computer equipment. Between 1957 and 1972, the National Science Foundation expended $85 million to support the purchase by more than 200 universities of computer hardware.

In an emergent discipline that depended on access to state-of-the-art equipment to conduct much of its research, these facilities grants for equipment literally laid the foundations

(and in other cases, provided the equipment that was placed on those foundations) for many universities' computer science departments. According to Norberg and O'Neill (1992), "there were virtually no formal programs" in computer science in U.S. universities as of 1959. By 1965, the Association of Computing Machinery (ACM) reported that more than 15 universities offered doctorates in computer science and 17 offered bachelor's degrees (ACM Curriculum Committee on Computer Science, 1965), and the output of degreeholders rapidly expanded. Facilities grants peaked in 1967 and began to decline thereafter as a paradoxical consequence of White House intervention to create an Office of Computing Activities within NSF, which assigned a higher funding priority to computer education than to facilities.

The National Science Foundation also supported academic research in software. Figure 2 displays the growth of NSF funding for research in software and related areas during 1956–80, which cumulatively amounted to more than $250 million (1987 dollars).[10] Among the contributions supported in part or entirely by Foundation grants were the development of PASCAL, pathbreaking work in principles of software engineering, and early object-oriented programming languages, such as CLU.

Figure 2 about here.

Apart from its facilities program, the support of the National Science Foundation for computer science research was

organized along classic "basic research" principles of peer review and individual support. Because or in spite of this structure, NSF support was widely dispersed among U.S. universities.[11] The distribution and consequences of NSF's R&D programs contrast with those of DARPA computer science funding.[12]

Rather than being spread among many universities, ARPA support was concentrated among leading U.S. research universities, primarily benefiting Carnegie-Mellon, MIT, Stanford, and the University of California at Berkeley.[13] ARPA funding was intended to support the long-term development of institutional and team strengths, and therefore was not allocated exclusively on the basis of individual performance or promise. Although expert panels played an important role in overseeing and reviewing its research programs, ARPA's support for computer science research was less tightly controlled by peer review than that of the National Science Foundation. This management approach was associated with high levels of flexibility and responsiveness to the needs of academic researchers.[14]

For much of ARPA's existence, its support for academic computer science research was channelled through IPTO, whose budget for fiscal 1965-85 is shown in Figure 3 (IPTO was disbanded in 1986). ARPA funding of academic computer science research contributed a number of important innovations in software and computer architecture, including computer time-sharing (based on a project begun at MIT in the early 1960s when campus demand for computer time began to outpace the available

supply), artificial intelligence architectures and software (including the LISP program), computer networking and communications (the ARPANET, forerunner of the NSFNET that underpins national and international electronic mail, was undertaken by ARPA as a means of linking researchers at its scattered "centers of excellence"),[15] and important modifications to the UNIX operating system to improve its performance in computer networking applications. As we discuss in more detail below, during the late 1970s and early 1980s, ARPA also undertook a major initiative to improve software development and maintenance practices.

Figure 3 about here.

In addition to its contributions to software innovations, ARPA's concentrated research funding made important educational contributions. ARPA research support for computer science in these institutions, major producers of academic researchers, had a substantial impact on computer science research and education elsewhere in U.S. universities. According to Norberg and O'Neill (1992, pp. 140–141), 26% of the faculty in the 40 leading U.S. university departments of computer science as of 1990 had received their computer science Ph.D. degrees from one of the three major universities supported by ARPA--Stanford, Carnegie-Mellon, or MIT. The influence of these three institutions was even greater among the top 10 U.S. computer science departments,

where 42% of the tenured faculty and 53% of the nontenured faculty had received their Ph.D.s from one of these three universities.[16] Other Ph.D. graduates of these universities also found employment in senior positions in firms such as Silicon Graphics, Microsoft, and Sun Microsystems. Many faculty at these institutions also entered into formal and informal consulting relationships with private firms, and others were directly involved in the foundation of such important hardware and software firms as the Carnegie Group, Ingress, and Thinking Machines.

The original aims of ARPA support for academic research in computer science were the creation of a basic research infrastructure in this new discipline, which was already being exploited by defense agencies for applications. This "infrastructure-building" goal incorporated support for training of personnel as well as for research. Its educational goals also influenced ARPA's academic research programs. Support for the development of computer time-sharing, for example, was motivated by concern over the ability of universities to train significant numbers of undergraduate and graduate students in computer science as much as by direct defense needs.

After the late 1960s, the mix of IPTO projects and funding shifted in favor of applications. This shift was a response to the growing budget of IPTO, the diminishing tolerance of Congressional and executive branch policymakers for fundamental research programs within the defense budget, and demands from the

uniformed services for near-term solutions to such challenges as software development and maintenance. The increase of $10.5 million (more than 35%) in the IPTO budget during fiscal 1971-75, for example, masked an absolute decrease in the most fundamental research areas and increases in support for more applied projects. According to Norberg and O'Neill (1992),

> the IPTO budget in FY 1971 represented 60% for basic research and 40% for exploratory development. By FY 1975, the numbers were approximately reversed: 43% for basic research and 57% for exploratory development. The split remained essentially the same for the FY 1980 budget, 42% and 58%, respectively. (pp. 8-9).

Defense-related support for academic computer science research continued to shift toward applied research through the mid-1980s. The effects of this shift were enhanced by the growth of defense-related R&D within overall federal computer science R&D funding. Expansion in this source of academic R&D funding, which was more development-oriented than federal R&D funding from other sources, tilted academic computer science research toward applications. Yudken and Simons (1988) argued that

> An increasing percentage of the nation's applied computer science research is being performed in universities and colleges. In 1987, academia received about 40% of all computer science federal obligations, an increase of approximately 25% since 1982. Academia's share of federal

> funding for applied computer science research has increased to 33% in 1987, up from 11.8% in 1982. However, its share of federal support for basic computer science research dropped from 78% to 66%. (pp. 62-63).

During the 1980s, ARPA began to develop a new "bridging" institution that was intended to facilitate and accelerate the movement of academic computer science research results, especially those relevant to software, into industry. The development of the SEI implied some shift in the relationship between academic research and defense-related software development. Support for university research alone no longer provided the necessary infrastructure for solution of serious problems in defense-related software management. Instead, a new organization was needed to conduct applied research and to support the dissemination and application within industry of the results of this and other research.

The Institute was intended by its sponsors to be sited in or near a research university, but it would provide a "halfway house" between academic research and application, supporting and accelerating the transfer of advances in software engineering and design from computer science research to application in commercial and defense-related firms.[17] The establishment of SEI appears to have reflected some concern (revealed as well in the shifting composition of Defense Department computer science R&D spending) that the returns to the large DoD investment in research were not being realized to a sufficient extent or with

sufficient speed in defense-related applications. The 1984 announcement of the formation of the Institute, located at Carnegie-Mellon University, projected a five-year DoD contribution to the SEI budget of $103 million. DoD funding for the Institute expanded from $5 million in 1985 to an annual contribution of almost $20 million during the late 1980s; since 1992, the annual DoD contribution has been reduced to roughly $15 million.

In contrast to the SAGE air defense system, a tightly targeted development project whose management was shifted by MIT to the semi-autonomous Lincoln Labs, the SEI focused on development and dissemination of generic tools and techniques for software engineering with defense applications. Its establishment seems to reflect some divergence between the mainstream of academic research in computer science and defense-related requirements for software innovations. Although U.S. universities developed a number of key innovations in computer hardware and software during the 1950s and 1960s, the establishment of the SEI suggests that the future relationship between these universities and defense-related software development may be less close.

4. Defense-related programs and the U.S. software industry

Defense-related procurement and R&D programs supported the growth of a number of postwar U.S. high-technology industries, including commercial aircraft, semiconductors, and computer

hardware for much of the postwar period. Although the computer software industry benefited from large DoD programs for R&D and procurement, the effects of these programs differed somewhat from other postwar high-technology industries. In the semiconductor industry, for example, DoD R&D programs produced few of the major technical advances embodied in commercial (as well as military) products (Tilton, 1971; Levin, 1982; Mowery and Steinmueller, 1994). Private firms accounted for numerous advances in software, but defense-related R&D programs, notably those of ARPA, supported important advances in fundamental knowledge of computer architecture, software languages, and design that found applications in both the civilian and defense sectors of the emergent industry.

Defense-civilian spillovers nevertheless were important in software. One basis for assessing their importance is provided by Flamm's tabulation of major advances in computer software during 1950-80 (1988, Table A-4). Of the 45 advances listed by Flamm as having originated in the U.S., the development of 18 was funded by the federal government, and all but one of these innovations drew on funding from the Pentagon and related military services. The prominent role of university-based research also is apparent from the fact that of these 18 innovations, 9 were developed in universities (including MIT's Lincoln Labs, the developer of the SAGE air defense computer system). The central place of universities in defense-civilian spillovers is unique to software among the postwar U.S.

industries characterized by high military R&D and procurement spending.

The spillovers identified in Flamm's tabulation assume two forms: (1) innovations that were first sold to federal agencies; and (2) innovations that were first sold to private firms. Most of the widely remarked defense-civilian technological spillovers in other industries and technologies fall into the first of these two categories. Fourteen of the 18 innovations in Flamm's tabulation that were developed with federal funds were first sold to federal agencies, while 4 of them were first applied outside of the federal government. Examples of the second category of defense-civilian spillovers in other postwar U.S. industries are rare, perhaps reflecting the emphasis of defense-related R&D in these other industries on specific mission applications.

A mere count of these spillovers says little or nothing about their economic significance. The innovations listed in Table 1, however, include a number of major advances. The compiler that was developed for the MIT Whirlwind, for example, contributed to the development of higher-order languages such as FORTRAN.[18] COBOL, which was described in 1972 as one of the two languages (the other being FORTRAN) that "...into the foreseeable future" would dominate "...most of the world's serious production programs" (Rosen, 1972, p. 591), is a high-order programming language developed to specifications formulated by a committee of industrial and military experts that was sponsored for much of its life by the Defense Department. DoD support for the

committee reflected military policymakers' growing concern over the costs and incompatibility of the rapidly expanding military investment in software for data-processing applications, as opposed to the "mission-critical" applications in weapons systems that inspired the development of the defense programming language of the 1980s, Ada.[19] Still another important spillover from defense to advances in civilian software technology, noted above, was the investment by the Defense Department during the 1950s in training the programmers who created the software for the SAGE air defense computer system.

Table 1: "Spillovers" in the U.S. Software Industry, 1950-75

Year	Innovation	Federally Funded?	1st Sale to Federal Agency?	Developer
unknown	Project Rye	Yes	Yes	Developed by Sperry Rand for National Security Agency.
early 1950s	APT language	Yes	Yes	MIT Whirlwind
1954	Whirlwind batch operating system	Yes	Yes	MIT Whirlwind
1957	SAGE time-sharing	Yes	Yes	MIT Lincoln Labs/System Development Corp.
1959	COBOL language	Yes	Yes	Developed to DoD specifications.
1959	Jovial	Yes	Yes	Developed by System Development Corp. for DoD.
1962	MIT time-sharing system	Yes	Yes	Developed at MIT with DARPA funding.
1963	Q-32 time-sharing system	Yes	Yes	Developed by System Corp. for DoD.
1963	Multiprocessor system	Yes	Yes	Developed by Burroughs for D-825 military computer.
1963	JOSS dedicated time-sharing system	Yes	Yes	Developed at RAND for DoD.
1964	Culler-Fried time-sharing system	Yes	Yes	Developed at TRW for DoD.
1966	Project Genie general time-sharing system.	Yes	Yes	Developed at U.C. Berkeley for Scientific Data Systems with DoD funding.
1968	MULTICS advanced time-sharing system	Yes	Yes	Developed at MIT with DARPA funding.

Table 1 (contd.)

Year	Innovation	Federally Funded?	1st Sale to Federal Agency?	Developer
1972	TENEX time-sharing system	Yes	Yes	Developed by BBN with DARPA funding.
1953	Algebraic compiler	Yes	No	Developed at Whirlwind at MIT.
1956	SHARE assembly pgm.	Yes	No	Developed at United Aircraft & distributed through IBM's SHARE program.
1963	Carnegie Tech. remote job entry system.	Yes	No	Developed at Carnegie Institute of Technology with DARPA funding.
1964	Basic	Yes	No	Developed by GE and Dartmouth with NSF funding.

SOURCE: Flamm (1988), Table A-4.

The interaction between defense and civilian applications and technological developments in the emergent U.S. software industry differed from that seen in other U.S. high-technology industries in the postwar period for at least two reasons. First, the share of defense-related demand within total software industry revenues remained high for a longer period of time than was true of such industries as semiconductors. According to Fisher (1978), annual DoD software expenditures amounted to $3-3.5 billion in 1973. Since total software industry revenues were no more than $4.2 billion in 1977 (Siwek and Furchtgott-Roth, 1993, p. 15), defense demand accounted for a substantial fraction of software industry revenues in the early and mid-1970s. As late as the early 1980s, some thirty years after the beginnings of software production, military demand may have accounted for 50% of total software industry revenues.[20] By contrast, only ten years after the commercialization of the integrated circuit in 1958, defense-related demand, which had accounted for 100% of industry shipments in 1962, amounted to 37% of the market (Mowery and Steinmueller, 1994, pp. 211-213).

Second, defense demand throughout the postwar period has been dominated by highly specialized custom and embedded software. This characteristic of military demand may well have reduced product-embodied "spillovers" (as opposed to spillovers based on defense R&D) of the type that were significant in other sectors. Since a large share of DoD R&D funding in computer science during much of the 1950s and 1960s focused on fundamental

R&D, rather than development, many of the military-civilian spillovers in software assumed a generic, rather than product-specific, character, and university research was central to their development. The divergence between the characteristics of products demanded by the military and those demanded in the commercial market that has affected such U.S. industries as semiconductors and aircraft thus may prove to be less significant in software, because this divergence has been a central factor from the industry's earliest days.

There exists no reliable time series of DoD expenditures on software procurement that employs a consistent definition of software, e.g., separating embedded software from custom applications or operating systems and packaged software, etc. The data on software expenditures in Figure 4 are also inconsistent in their treatment of DoD expenditures on software maintenance, as opposed to procurement. Nevertheless, the trends in these data are dramatic--in constant-dollar terms, DoD expenditures on software increased more than thirtyfold in just over 25 years, from 1964-90. Throughout this period, DoD software demand was dominated by custom software, and DoD and federal government demand for custom software accounted for a substantial share of the total revenues in this segment of the U.S. software industry. Much of the rapid growth in custom software firms during the 1969-80 period that is discussed in Chapter 2 reflected expansion in federal demand, which in turn was dominated by DoD demand.[21]

Figure 4 about here.

This rapid growth in DoD software expenditures, coupled with other developments in DoD programs and in the structure of the U.S. software industry, gave rise to concern within the Pentagon over "productivity bottlenecks" in software production. Software support and maintenance, i.e., changing programs to adapt to new mission requirements, eliminate errors, or improve performance, grew rapidly as a share of total software and hardware costs. By 1985, software support alone was estimated to account for at least 50% of the cost of complex defense computer systems, a significant increase from its original share of less than 10% in the early 1960s (Defense Systems Management College, 1990, p. 2-3). Policymakers also worried about the availability of skilled software engineers and the ability of any technical staff to maintain the rapidly growing, aging, and extremely heterogeneous installed base of software in DoD weapons systems. The Strategic Defense Initiative, with its requirements for large amounts of highly reliable, error-free software, made all of these problems more acute and visible.[22] These concerns led to a series of initiatives in the 1980s, including the development of the Ada high-order language for defense applications, expanded programs in software engineering that included the STARS (Software

Technology for Adaptable, Reliable Systems) program, and the Software Engineering Institute.

The complexities of DoD software procurement and maintenance were exacerbated by the importance of "embedded" software, contained in instruments or in components of larger weapons systems. This type of software accounted for more than 55% of total DoD software expenditures in 1973,[23] and its share of DoD's total software budget may well have increased since then. Embedded software brought with it considerable costs and benefits. Software enabled much greater flexibility, and often much greater speed, in modifying deployed weapons systems for new missions. The 1982 Joint Services Task Force on Software Problems estimated that modifying the capabilities of the Air Force F-111 aircraft through software rather than hardware enabled a fiftyfold savings in cost and a threefold acceleration in the deployment of the modified aircraft.[24] This "mission-critical" embedded software also had to meet requirements for reliability and quality control that were far more demanding than those associated with conventional data processing operations.

Perhaps the greatest cost associated with its widespread use was the fact that most of the embedded software employed in weapons systems developed before 1982 was specific to a given weapons system or contractor, and a lack of standards implied a lack of compatibility.[25] Moreover, as several other chapters in this volume point out, the dominance of custom applications

within military software minimized incentives to create generic tools or languages:

> There has been little incentive for individual projects to expend the effort and resources necessary to provide facilities that would be generally useful, especially when there are few, if any, other projects using the same programming language. This may also account for the lack of off-the-shelf software...At least 450 general-purpose programming languages and (incompatible) dialects are used in DoD embedded computer applications--and none is widely used. (Fisher, 1978, p. 26).

Maintenance and support of defense software were especially difficult, since idiosyncratic programs for specific systems or applications, developed with limited documentation, could remain in service for years or even decades.

When confronted with a similar problem in 1959, DoD had supported an industry-led committee that laid out the requirements and specifications for a higher-order language, COBOL, that was developed by private firms. In response to similar confusion in "mission-critical" software, the Defense Department launched a major effort to develop an "official" standard for its software procurement in 1974. But in 1974, the project was controlled more tightly by DoD, which appointed the committee charged with defining the requirements and evaluating the suitability of existing languages to meet them. In the absence of a satisfactory language, a competition was held to

evaluate competing designs of a new language. The result, developed largely by Honeywell-Bull on a DoD contract (another contrast with the COBOL experience), was the Ada language, announced in 1981 and required in all major DoD procurement programs.

The Ada initiative was an effort by DoD to create a standardized software environment that would create a "virtuous cycle" similar to that associated with the growth of a "dominant design" in the civilian microcomputer market, in which the diffusion of the IBM PC supported growth in the production of low-cost packaged software for a huge variety of applications (See Chapters 2 and 5 in this volume). In contrast to COBOL, Ada has not been extensively employed thus far in nondefense systems, partly because it was developed to meet requirements that had few civilian counterparts.[26]

The Ada initiative was joined in the early 1980s by a broader effort to enhance the efficiency of defense-related software development and procurement that led to increased DoD funding for generic software engineering research and related activities. Beginning in the early 1980s, with the report of the Joint Services Task Force on software development and procurement, a succession of studies[27] reviewed DoD software policies and agreed on three goals: (1) the costs of software procurement and maintenance must be brought under control; (2) one means to achieve this goal was through greater exploitation of the resources and products of the civilian software industry

(so-called "COTS"--commercial off-the-shelf software) for many defense-related software needs;[28] and (3) DoD funding for expanded research on and dissemination of software engineering techniques provided one means to achieve the first two goals. In contrast to the Ada initiative, which defined a defense-specific set of requirements that produced a "dedicated" DoD high-order computer language, these efforts of the 1980s attempted to bridge the gap between defense and civilian technological developments and "unify" the civilian and defense industrial base in software. Another motive for efforts to link the civilian and defense-related software industries more closely was growing concern by IPTO and ARPA managers with the international competitiveness of the U.S. computer and electronics industry complex. This concern motivated large programs in "Strategic Computing," which included expanded research in software development.

As originally planned, the STARS program and the Software Engineering Institute had a hardware complement in the Very High Speed Integrated Circuit (VHSIC) program, which like these software initiatives sought to exploit civilian technological capabilities in the semiconductor industry for defense-related applications (Martin, 1983). The STARS program was intended to develop better methods for defining software requirements and specifications in a flexible manner that would also enhance re-use of software code. Among STARS' goals were computer-aided software engineering tools for developing Ada and other software. In December 1989, ARPA shifted the STARS program to increase the

involvement of commercial software vendors in the development of techniques that drew more heavily on civilian software products and could be sold in civilian, as well as military markets. This shift in program philosophy was associated with new requirements that STARS contracts involve commercial software vendors.[29]

The combined effects of sharp cuts in the SDIO budget and across-the-board reductions in overall defense spending after 1989 reduced defense-related R&D spending in software, even as civilian agencies such as the National Science Foundation increased their computer science research budgets. The defense share of federal computer science R&D funding declined from almost 60% in fiscal 1986 to less than 30% in fiscal 1990 (Clement, 1987, 1989; Clement and Edgar, 1988), and by the early 1990s, defense demand accounted for a declining share of industry markets.

Although the past development of the U.S. software industry exhibits a pattern of military-civilian interaction that contrasts with that of other U.S. high-technology industries, the present relationship between civilian and military software technology appears to resemble that of other industries. Defense-related demand accounts for a declining share of industry output, and technologies developed for civilian applications appear to promise higher performance at lower cost. In response, the Defense Department has attempted to strengthen its links with the commercial sector of the software industry, just as it now

seeks to do in such products as flat-panel displays (Davis and Zachary, 1994).

The success of these efforts, however, remains very uncertain. One of the most important impediments to the development of such links, defense contracting policies on the ownership of code, has scarcely been addressed (Zraket, 1992, pp. 310-311). The Ada initiative defined a set of requirements and an entire language that has thus far produced few "spillovers" or "spin-on" benefits for the civilian software industry, and few new entrants have been attracted to the military software market. The creation of a separate institution for the development and dissemination of software engineering techniques for defense-related applications creates some risk that, despite the intentions of its sponsors, the Software Engineering Institute may contribute to further divergence between defense and nondefense software development techniques and products.

The area of software engineering also reveals divergence between defense and commercial technologies. DoD's conceptualization of the "software bottleneck" problem has until recently focused on techniques for design and management of the development of large-scale software systems that require very low error rates in code in organizations resembling "software factories" (SAGE was one prototype). Originally developed by SAGE contractor SDC, the software factory sought to increase productivity and reduce errors within the development

organization by systematically reusing parts of code on similar but not identical large-scale projects.[30]

The software factory and related techniques of software engineering remain relevant to the creation of complex, customized defense software, especially embedded, "mission-critical" software that cannot tolerate errors in code. But within the commercial software industry, standardization of platforms and languages, rather than code reuse, has been the key to great increases in efficiency and profitability. Many of the techniques of the software factory are unnecessary for mass-market, packaged software, early releases of which often are riddled with errors. In the commercial sector, where the problems of sharing code across (proprietary) organizations are serious, object-oriented programming may provide a way to share and reuse code in new ways and more effectively (Lavoie et al., 1992). But DoD has not pursued object-oriented techniques for software development. Ada is not an object-oriented language,[31] and its specificity to DoD applications means that it may not attract private developers' investment and effort in competition with far more widely used commercial operating systems and languages.

Without a radical shift in DOD's underlying weapons design and procurement philosophy, from one that emphasizes performance above all else to one stressing the use of standard hardware components and platforms, the military efforts to exploit commercial software are likely to remain ineffective. The case

of software suggests that the development of closer links between the defense-related and civilian sectors of high-technology U.S. industries will be very difficult and will take considerable time.

Our discussion of federal policy extends only through 1990, but developments since that date nonetheless merit a brief comment. Since 1992, ARPA has explicitly followed a policy (which, as we noted above, has been articulated within the agency since at least 1980, but rarely stated publicly) in which DoD and other federal agencies will support projects that have both commercial and defense applications (Alic et al. 1992; Bingaman and Inman 1992). Although ARPA's fundamental R&D support in software produced important advances in both civilian and defense application, there are ample gounds for skepticism about the possibilities for such "spin-on" benefits from development funding. As we noted earlier in this chapter, the differences between defense and commercial requirements and markets in software remain so great that genuinely dual-use benefits from technology development spending are likely to be rare.

5. NASA Software Programs

Another federal agency with significant software-related activities is the National Aeronautics and Space Administration (NASA), which required complex flight-operations software (both on the spacecraft and on the ground) for its manned space exploration missions and embedded software for its unmanned

planetary satellites. As Figure 5 (from the Defense Systems Management College, 1990, p. 7-2) shows, NASA's software requirements for the Space Shuttle and other manned spaceflight missions were more complex (measured in terms of the number of instructions) than any single U.S. weapons system, including the B-1 bomber or the AWACS airborne air defense radar system. These demanding, mission-specific requirements forced NASA and its civilian contractors to develop advanced techniques of software engineering.

Figure 5 about here.

In the manned spaceflight program, NASA's first software contractor was MIT's Instrumentation Laboratory (later known as Draper Labs), which was chosen by NASA on the strength of its performance in developing guidance systems for the Polaris nuclear missile (Tomayko, 1988). Although the Apollo software program was eventually successful, the enormous difficulties associated with the effort led NASA to seek an on-site contractor in later programs for the development of flight operations and ground-based control software.[32]

The demanding space and weight requirements of NASA's manned and unmanned spaceflight missions meant that most of the on-board computer hardware and software was unique to these missions. As a result, the software developed for on-board applications yielded relatively few "spillovers" to commercial applications.

In the case of ground-based computer systems to manage the extremely complex tasks of launch and communications, however, "off-the-shelf" hardware was exploited far more extensively. Moreover, many of the techniques of software development for these applications were employed in broader commercial markets by the primary vendor, IBM's Federal Systems Division (now known as the Federal Systems Company).[33] IBM, which located a large software development facility near the Johnson Space Center in Houston, was the major supplier of software for ground control systems throughout NASA's manned spaceflight program, and later became the prime contractor for the Space Shuttle's software. Beginning with the Mercury program, IBM's development of computer systems and software for ground control applications yielded important commercial spillovers:

> For IBM and NASA, the development of the Mercury control center and the network was highly profitable...Large central computers with widely scattered terminals, such as airline reservation systems, have their basis in the distant communications between Washington and a launch site in Florida. (Tomayko, 1988, p. 248).

Other important architectural advances spurred by IBM's experience as the prime contractor for ground-control hardware and software include the demonstration of the design principles that underpinned the subsequent innovation of virtual memory, as well as major improvements in IBM's internal software engineering practices and guidelines.[34] All of these yielded important

commercial spillovers for IBM and for the broader U.S. software industry.

Although they were valuable for the development of complex, custom software programs, the software engineering advances supported by IBM's NASA experience nevertheless are less relevant to the packaged software market that became so important during the 1980s, as Smith and Cusumano (1993) point out:

> The [IBM NASA software development] process is an excellent fit for the environment: a dedicated customer, a limited problem domain, and a situation where cost is important but less of a consideration than zero defects. For the wide range of commercial software developers that do not operate in this type of environment, the...complete FSC Houston approach is not feasible, although variations of the process are clearly possible and used at other IBM sites and other companies...Drawing upon this success in process improvement and quality delivery, the IBM Federal Systems Company has also created a team that now goes out and consults on the software development process. (Smith and Cusumano, 1993, p. 19).

In spaceflight no less than in defense-related procurement large government software contracts now appear to yield fewer benefits for vendors in the commercial market, underlining the divergent requirements and strategies associated with these sectors of the U.S. software industry. IBM's 1993 divestiture of its Federal Systems Company seems to underline the limited

relevance of the large-scale software factory for developing commercial software (which accounted for $11 billion in 1993 corporate revenues, according to the firm's Annual Report).

6. Conclusion

The federal government's role in the development of the U.S. software industry is broadly similar to its role in the development of such other postwar high-technology industries as semiconductors, computer hardware, and commercial aircraft. In all of these cases, federal expenditures on R&D and procurement were motivated primarily by defense concerns in the context of the Cold War. Defense-related expenditures produced important "spillovers" for commercial applications, which was also the case in software. Especially in semiconductors and computer hardware, federal procurement also supported significant entry by many startup firms, as in the software industry.

But the apparent similarities between software and other postwar "dual-use" industries mask some important differences in the structure of federal policy toward the software industry. For example, throughout the brief history of the software industry, defense-related demand for (largely custom) software has accounted for a much larger share of the total market than was true of semiconductors. Federal funding of university research and development activities appear to have been more important to the evolution of this postwar industry than is true of semiconductors or aerospace. In contrast to these industries,

which drew on established academic disciplines even as they transformed them, the software industry relied on the creation of a new academic discipline, computer science. Federal policymakers in agencies such as ARPA focused their R&D support on universities because of the need for a new academic infrastructure of training and research for the development of a technology with numerous defense-related applications. Moreover, the explicit "targeting" of institutional strength in computer science, combined with the defense mission of agencies such as ARPA, meant that this source of academic research support employed policies and criteria for support that contrasted with the peer review system more commonly associated with federal support for fundamental academic research. The contrasting postwar histories of the British and U.S. computer industries appear to stem in part from the very different policies adopted by each nation's military establishment to the support of university research and education in computer science.

The National Science Foundation also included support for infrastructure through its funding of computer purchases by U.S. universities, and thereby complemented the focused policies of ARPA. The important role of universities within the software industry, and the importance of federal financial support for the research facilities of these universities, both suggest some interesting similarities between the U.S. software and biotechnology industries. In the case of biotechnology, of course, the National Institutes of Health have played an

indispensable role in supporting fundamental research and equipment acquisition that have yielded major commercial applications.

Although ARPA's R&D support focused on specific areas of opportunity in computer science, the overall structure of federal R&D support in software-related fields was not tightly "targeted" on specific civilian applications, or even on civilian technology development. In contrast to the ambitious programs mounted within Europe and Japan, U.S. government R&D policy devoted relatively little attention (until recently) to civilian applications. Indeed, the recent federal emphasis on technology development and civilian applications indicate some shift in federal policy in the direction of European and Japanese programs, which have had little success in the software industry.

The history of federal policy in the software industry supports a strong role for public funds in the creation of a research infrastructure, including support for the production of trained personnel, rather than a policy that attempts to focus research programs on the development of specific technologies for civilian applications. It also suggests the importance of support for institutions and facilities, in addition to individual investigators, in order to spur the growth of new academic disciplines. In all of these respects, the lessons of postwar federal policy in the software industry closely resemble those of federal technology policy in other sectors (See Nelson, 1985; Mowery and Rosenberg, 1989). Paradoxically, the national

security rationale for much of the DoD and ARPA funding, especially the ARPA funding of university "centers of excellence," may have insulated these programs, which did not operate solely via peer review, from the distributional politics that otherwise might have forced the use of very different criteria for allocation.[35] The lack of a "civilian competitiveness" rationale for them may have increased the contribution of these federal R&D programs to the U.S. software industry's competitiveness.

Although military–civilian spillovers were important in the software industry, their structure appears to have differed somewhat from those associated with other postwar dual-use industries. In a number of other postwar industries, as we noted in the introduction to this chapter, government R&D support and defense-related procurement often yield commercial applications in the commercial sector during the earliest stages of a technology's development. With the maturation of the technology and the emergence of a commercial industry, however, government procurement needs diverge from those of the commercial market, especially if this market becomes large relative to the government market (See Mowery and Rosenberg, 1989; Cowan and Foray, 1994). This development often reduces the spinoffs from defense-related government spending on R&D and procurement, and military applications may come to rely more heavily on “spinons” from the commercial sector.

This transition seems to have occurred quite early in the development of the U.S. software industry. The early experimental machines at universities, such as Whirlwind or the IAS computer, yielded a number of generic software concepts and tools. But very soon the private sector, notably IBM, began volume production of standard platforms. At the same time, the commercial sector, responding to the resulting growth in the market for standard commercial applications, began to provide generic programming tools and languages. Private firms also extended some financial support for computer science and developed end-user capabilities through user groups. The mid-1950s had some interesting similarities with the current situation in the software industry, with standardized hardware platforms supporting the growth of commercial production of software, although independent software vendors now play a much more prominent role in the commercial sector (see chapter 2).

Rapid growth in defense-related demands for software, and for science and technology in general, during the 1950s expanded government spending on software and computer science. Government funding stimulated the creation of a university-based infrastructure for the development "generic" technology and abstract principles, many of which were applicable to both military and commercial software. Rather than applications of technologies developed for defense purposes, software-related spillovers frequently flowed from defense-related support for fundamental research, and universities were important sources of

such spillovers. Had private firms retained the primary responsibility for the "legitimation" of computer science, this process might well have taken considerably longer and might have restricted the diffusion of the results of university-based research.

The divergent nature of military and commercial demand for software has not been significantly reduced by the development of Ada, and it is likely to hamper efforts to improve military utilization of commercial software products. Indeed, the shift in defense-related R&D support toward applied research may reduce the future "spillovers" to commercial appplications formerly generated by this research funding. The establishment of the Software Engineering Institute, which was intended to supplement the role of U.S. universities in supporting defense-related software R&D, suggests that this divergence may also affect the relationship between DoD R&D programs and academic research in software and computer science. The contrasting roles of MIT in establishing Lincoln Labs to manage the SAGE project and ARPA in the establishment of the SEI highlights this changing relationship.

The political and economic circumstances within which these federal agencies influenced the early development U.S. software industry were in many respects unique. Although federal support for its university infrastructure will remain vital, the software industry has achieved sufficient scale and economic vitality that federal R&D policy is likely to exercise less direct influence

over its future technological development. But the lessons of federal policy in this industry, at least some of which seem to contradict the spirit of current federal policy initiatives in software and other high-technology industries, remain relevant in the U.S. economy of the future.

Bibliography

Alic, J.A., L.W. Branscomb, H.A. Brooks, and A. Carter, Beyond Spinoff (Boston: Harvard Business School Press, 1992).

Arrow, Kenneth J., "Economic Welfare and the Allocation of Resources to Invention," in Richard R. Nelson, ed., The Rate and Direction of Inventive Activity: Economic and Social Factors (Princeton: Princeton University Press, 1962).

Aspray, William, John von Neumann and the Origins of Modern Computing (Cambridge: MIT Press, 1990).

Aspray, William, and Bernard O. Williams, "Computing in Science and Engineering Education: The Programs of the National Science Foundation," presented at the IEEE Electro/93 conference, Edison, NJ, April 27-29, 1993a.

Aspray, William, and Bernard O. Williams, "The National Science Foundation's Computer Science Research Program," unpublished MS., 1993b.

Aspray, William, and Bernard O. Williams, "Arming American Scientists: The Role of the National Science Foundation in the Provision of Scientific Computing Facilities for Colleges and

Universities," forthcoming, Annals of the History of Computing, 1994.

Bach, G.L., "A Computer for Carnegie," Annals of the History of Computing 8, 1986, 39-41.

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

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

Benington, H.D., "Production of Large Computer Programs," Annals of the History of Computing 5, 1983, 350-361.

Bingaman, Jeff, and Bobby R Inman, "Broadening Horizons for Defense R&D," Issues in Science and Technology 9, Fall 1992, 80-85.

Boehm, B., and T.A. Standish, "Software Technology in the 1990s: Using an Evolutionary Paradigm," IEEE Computer, 1983,

Brosgol, B.M., "Ada," Communications of the ACM 35, November 1992, 41-42.

Burgess, Angela, "DOD Budget Embodies New Acquisition Plan," IEEE Software 9, May 1992, 99.

Clement, J.R.B., "Computer Science and Engineering Support in the FY 1988 Budget," in Intersociety Working Group, ed., AAAS Report XII: Research & Development, FY 1988 (Washington, D.C.: American Association for the Advancement of Science, 1987).

Clement, J.R.B., "Computer Science and Engineering Support in the FY 1990 Budget," in Intersociety Working Group, ed., AAAS Report XIV: Research & Development, FY 1990 (Washington, D.C.: American Association for the Advancement of Science, 1989).

Clement, J.R.B. and D. Edgar, "Comptuer Science and Engineering Support in the FY 1989 Budget," in Intersociety Working Group, ed., AAAS Report XIII: Research & Development, FY 1989 (Washington, D.C.: American Association for the Advancement of Science, 1988).

Cowan, R., and D. Foray, "Quandaries in the Economics of Dual Technologies and Spillovers from Military to Civilian Research and Development," unpublished MS, 1994.

Cusumano, Michael, Japan's Software Factories: A Challenge to U.S. Management (Cambridge: MIT Press, 1991).

Davis, B., and G.P. Zachary, "Electronics Firms Get Push from Clinton to Join Industrial Policy Initiative in Flat-Panel Displays," Wall Street Journal, 4/28/94, p. A16.

Defense Systems Management College, U.S. Department of Defense Mission Critical Computer Resources Management Guide (Washington, D.C.: U.S. Defense Department, 1990).

Feldman, M.B., "Ada Experience in the Undergraduate Curriculum," Communications of the ACM 35, November 1992, 53-67.

Fisher, D.A., "DoD's Common Language Programming Effort," IEEE Computer 11, March 1978, 24-33.

Fisher, Franklin M., James W. McKie, and Richard B. Mancke, IBM and The U. S. Data Processing Industry (New York: Praeger, 1983).

Flamm, Kenneth, Targeting the Computer (Washington, DC: The Brookings Institution, 1987).

Flamm, Kenneth, Creating the Computer (Washington, DC: The Brookings Institution, 1988).

Flamm, Kenneth S., and Gary L. Denman, "Testimony before the Subcommittee on Defense," Committee on Appropriations, U. S. House of Representatives, April 13, 1994.

Gries, D., R. Miller, R.Ritchie, and P. Young, "Imbalance Between Growth and Funding in Academic Computer Science: Two Trends Colliding," Communications of the ACM 29, 1986, 870-878.

Hendry, J., Innovating for Failure (Cambridge: MIT Press, 1990).

IBM Corporation, 1993 Annual Report.

Katz, Barbara, and Almarin Phillips, "The Computer Industry," in Richard R. Nelson, ed., Government and Technical Progress: A Cross-Industry Analysis (New York: Pergamon Press, 1982).

Knuth, D.E., and L.T. Pardo, "The Early Development of Programming Languages," in N. Metropolis, J. Howlett, and G.-C. Rota, eds., A History of Computing in the 20th Century: A Collection of Essays (New York: Academic Press, 1980).

Lavoie, Don, Howard Baetjer, William Tulloh, and Richard Langlois, Component Software: A Market Perspective on the Coming Revolution in Software Development, Special Research Report, Patricia Seybold Group, Boston, April, 1992.

Leslie, S., The Cold War and American Science (New York: Columbia University Press, 1993).

Levin, Richard C., "The Semiconductor Industry," in Richard R. Nelson, ed., Government and Technical Progress: A Cross-Industry Analysis (New York: Pergamon Press, 1982).

Levine, Bernard, "Largest Funds Emerge So far for TRP Wins," Electronic News, February 28, 1994 p. 1.

Martin, E.W., The Context of STARS," IEEE Computer, November 1983, 14-17.

Mowery, David C., and Nathan Rosenberg, "Government Policy and Innovation in the Commercial Aircraft Industry, 1925-1975," in Richard R. Nelson, ed., Government and Technical Progress: A Cross-Industry Analysis (New York: Pergamon Press, 1982).

Mowery, David C., and Nathan Rosenberg, "The U.S. National System of Innovation," in R.R. Nelson, ed., National Innovation Systems: A Comparative Analysis (New York: Oxford University Press, 1993).

Mowery, David C., and W.E. Steinmueller, "Prospects for Entry by Developing Countries into the Global Integrated Circuit Industry: Lessons from the United States, Japan, and the NIEs, 1955-1990," in D.C. Mowery, Science and Technology Policy in Interdependent Economies (Boston, MA: Kluwer Academic Publishers, 1994).

National Science Foundation, Office of Computing Activities, "Director's Program Review: December 15, 1970," unpublished document, Program Review Office, National Science Foundation, Washington, D.C., 1970.

Naur, P., B. Randell, and J.B. Buxton, eds., Software Engineering Concepts and Techniques: Proceedings of the NATO Conference (New York: Petrocelli/Charter, 1976).

Norberg, A.L., and J.E. O'Neill, with contributions by K.J. Freedman, A History of the Information Processing Techniques Office of the Defense Advanced Research Projects Agency (Minneapolis: Charles Babbage Institute, 1992).

Phillips, C.A., "Reminiscences (Plus a Few Facts)," Annals of the History of Computing 7, 1985, 304–313.

Redmond, Kent C., and Thomas M. Smith, Project Whirlwind: History of a Pioneer Computer (Bedford, MA: Digital Press, 1980).

Rosenberg, Nathan, Perspectives on Technology (New York: Cambridge University Press, 1976).

Rosenberg, N., "Scientific Instrumentation and University Research," Research Policy 21, 1992, 381–390.

Rosenberg, N., and R.R. Nelson, "American Universities and Technical Advance in Industry," Research Policy 23, 1994, 323-348.

Sammet, J.E., "Brief Summary of the Early History of COBOL," Annals of the History of Computing 7, 1985, 288-303.

Sapolsky, H., Science and the Navy (Princeton, NJ: Princeton University Press, 1990).

Siwek, S.E., and H.W. Furchtgott-Roth, International Trade in Computer Software (Westport, CT: Quorum Books, 1993).

Smith, S.A., and M.A. Cusumano, "Beyond the Software Factory: A Comparison of 'Classic' and PC Software Developers," Sloan School of Management working paper #96-93, 1993.

Stix, Gary, "Objective Data: DARPA Nudges Development of Object-Oriented Data Bases," Scientific American 266, March 1992, 108.

Tilton, J.E., The International Diffusion of Technology: The Case of Semiconductors (Washington, D.C.: The Brookings Institution, 1971).

Tomayko, J.E., Computers in Spaceflight (Washington, D.C.: National Aeronautics and Space Administration, 1988).

Tropp, H.S., ed., "A Perspective on SAGE: A Discussion," Annals of the History of Computing 5, 1983, 375-398.

U.S. Congress, Office of Technology Assessment, SDI: Technology, Survivability, and Software (Washington, D.C.: U.S. Government Printing Office, 1988).

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

U.S. Department of Defense, Joint Service Task Force on Software Problems, Report of the DOD Joint Service Task Force on Software Problems (Washington, D.C.: U.S. Department of Defense, 1982).

U.S. House of Representatives, Commitee on Science, Space, and Technology, Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation (U.S. Government Printing Office, 1989).

von Neumann, John, "First Draft of a Report on the EDVAC," 1945; reprinted in William Aspray and Arthur Burks, eds., Papers of John von Neumann on Computing and Computer Theory (Cambridge: MIT Press, 1987).

Wildes, Karl L., and Nilo A. Lindgren, A Century of Electrical Engineering and Computer Science at MIT, 1882-1982 (Cambridge: MIT Press, 1985).

Yudken, J.S., and B. Simons, "Computer Science Research Funding: Issues and Trends," Abacus, 1988, 60-66.

Zraket, Charles A., "Software: Productivity Puzzles, Policy Challenges," in John A. Alic, et al., eds., Beyond Spinoff (Cambridge: Harvard Business School Press, 1992), pp. 283-313.

NOTES

1. We are grateful to Jay Adams, William Aspray, Bruce Bruemmer, Randy Katz, Larry Druffel, Bonnie Chen, George Mazuzan, Scott Wallace, and Andrew Goldstein for invaluable assistance in the preparation of this chapter. Support for the second author's research for this chapter was provided by the U.S.-Japan Industrial Technology Management Program, sponsored by the Air Force Office of Scientific Research; by the Center for Research in Management of the Haas School of Business; and by the Alfred P. Sloan Foundation.

2. The Whirwind's cost of $3 million substantially exceeded the average cost of $650,000 for the other systems described below (Redmond and Smith, 1980).

3. "Prior to this system [the 650], universities built their own machines, either as copies of someone else's or as novel devices. After the 650, this was no longer true. By December 1955, Weik reports, 120 were in operation, and 750 were on order. For the first time, a large group of machine users had more or less identical systems. This had a most profound effect on programming and programmers. The existence of a very large community now made it possible, and indeed, desirable, to have common programs, programming techniques, etc." (Goldstine, 1972, p. 331).

4. Goldstine, one of the leaders of the wartime project sponsored by the Army's Ballistics Research Laboratory at the University of Pennsylvania that resulted in the Eckert-Mauchly computer, notes that "A meeting was held in the fall of 1945 at the Ballistic Research Laboratory to consider the computing needs of that laboratory 'in the light of its post-war research program.' The minutes indicate a very great desire at this time on the part of the leaders there to make their work widely available. 'It was accordingly proposed that as soon as the ENIAC was successfully working, its logical and operational characteristics be completely declassified and sufficient be given to the machine...that those who are interested...will be allowed to know all details.'" (1972, p. 217). Goldstine is quoting the "Minutes, Meeting on Computing Methods and Devices at Ballistic research Laboratory, 15 October 1945 (note 14). Flamm (1988), pp. 224-226, makes a similar point with respect to military attitudes toward classification of computer technology.

5. "The grants were in IBM's interest, because the corporation felt a strong concern with supporting and maintaining a close relationship with universities, and because an entire generation of students and faculty would associate computers and computing with `IBM.'" (Galler 1986, p. 37.) Fisher et al (1983, p. 169)

suggest, however, that this hoped-for "lock in" effect might in fact have been illusory.

6. In the case of MIT, IBM donated a model 704 computer in 1957, which was available free of charge to MIT seven hours a day and to 24 other New England universities another seven hours a day. IBM itself used the remaining 10 (nighttime) hours. (Wildes and Lindgren 1985, pp. 336-7.)

7. The IBM educational allowance program began in October 1955, with 60 per cent reductions in lease rates to universities. In May 1960, IBM changed the allowance to 20 per cent for administrative use and 60 per cent for academic use. In 1963, the company abandoned the administrative/academic distinction and reduced all allowances to 20 per cent on new orders. In 1965, IBM set up a sliding scale of allowances on the new 360 series, ranging from 20 per cent on the base model to 45 per cent on a high-end system. By 1969, the allowance had been reduced to 10 per cent. (Fisher et al. 1983, p. 172.)

8. According to some accounts (Baum, 1981), the Rand group got the programming job only after MIT, IBM, and AT&T had all declined it. IBM, for example, was concerned about how it would employ some 2,000 programmers once the project ended.

9. "Indeed, despite what was in many respects a first-rate network of contacts, the NRDC [National Research and Development Corporation] was not even aware of some of the military computer developments taking place in the 1950s and early 1960s. Nor were the people carrying out these developments in many cases aware of work on the commercial front. In America, in contrast, communications between different firms and laboratories appear to have been very good, even where classified work was involved." (Hendry, 1992, p. 162).

10. The data in Figure 2 are taken from Aspray and Williams (1993b), and include all NSF research funding in numerical analysis, computer theory, architecture, theoretical computer engineering, graphics, software, artificial intelligence, databases, and communications. I am indebted to Professor William Aspray and Dr. Andrew Goldstein of the IEEE Center for the History of Electrical Engineering for these data.

11. "...NSF wanted to develop a broad academic base for science education and scientific research. Thus it supplied facilities not only to the top research universities, but also to other universities liberal arts colleges, junior colleges, and even some high schools. While NSF was building a broad national infrastructure for science, DARPA was constructing a high-powered experimental computer science research program." (Aspray and Williams, 1994, p. 28).

12. This account draws on the excellent history of DARPA's Information Processing Techniques Office (IPTO) by Norberg and O'Neill (1992).

13. A 1985 study by an ACM committee found that average DoD funding per faculty member in the computer science departments of these four universities was $279,000; this amount dropped to $42,000 per faculty member in departments ranked 5-12. By contrast, NSF funding per faculty member in the top four departments was $31,000, an amount that rose to $46,000 per faculty member in departments ranked 5-12 and stood at $41,000 per faculty member in departments ranked 13-24 (Gries, Miller, Ritchie, and Young, 1986, p. 878).

14. "IPTO strove to develop a few centers of excellence in order to stimulate the computer science field in substantial ways, rather than by making a large number of small contracts. Contracts written for this purpose were brief. They specified in general ways the nature of the research to be pursued, the equipment to be purchased if needed, and the length of time of the contract....Manager rather than peer review made possible this rapid evaluation of proposals and issuance of an intent to contract with an institution." (Norberg and O'Neill, 1992, pp. 122–123).

15. The ARPANET packet–switching architecture and supporting software were developed by Bolt, Baranek, and Newman, an engineering firm that had "spun off" from MIT, rather than solely by academic researchers. Nonetheless, academic computer science researchers were very active participants in ARPA's early development of requirements for what became the ARPANET, as well as in the testing of the system. In 1989, ARPA turned over the network technology to NSF, which became the lead agency of an Executive Branch consortium guiding the future of the Internet (Turner 1989).

16. These data omit data for Ph.D. production from the University of California at Berkeley, which for much of this period did not grant Ph.D.s in computer science (a division of the university's

department of electrical engineering), although it was a major recipient of ARPA funding for computer science research. By virtue of this omission, these data understate somewhat the "true" influence of the top four ARPA-funded research universities in computer science.

17. "An institute like the SEI could play several influential roles. It could collect and integrate existing tools into common, unified, software life-cycle support frameworks. Such an institute could act as an integrative agent by energetically soliciting community opinion and helping the community to achieve a consensus on critical new suspects of shared infrastructure for the 1990's. Additionally, it could furnish effective institutional support for the technology insertion process. This process needs to be carefully planned and managed, and if it is not vigorously supported, an essential part of the overall job cannot be accomplished.

"A key feature of technology transfer is to have people from the DoD, industry, and academia rotating through the institute. Such rotation would have the additional benefit of keeping the institute fresh and vital over time. It would thus be a magnet for top talent without being a talent sink." (Boehm and Standish, 1983, p. 34).

18. Moreover, the Office of Naval Research supported considerable information exchange, workshops, and informal cross-fertilization that enabled the compiler, first developed by Laning and Zierler at MIT, to be quickly exploited by John Backus of IBM in creating FORTRAN: "During the first part of 1954, John Backus began to assemble a group of people within IBM to work on improved systems of automatic programming...Shortly after learning of the Laning and Zierler system at the ONR meeting in May, Backus wrote to Laning that 'our formulation of the problem is very similar to yours: However, we have done no programming or even detailed planning.' Within two weeks, Backus and his co-workers Harlan Herrick and Irving Zierler visited MIT in order to see the Laning-Zierler system in operation. The big problem facing them was to implement such a language with suitable efficiency...

"By November 1954, Backus's group had specified 'The IBM Mathematical FORmula TRANslating system, FORTRAN." (Knuth and Pardo, p. 241).

19. According to Phillips (1985), "In early 1959 there were 225 computer systems in the [DoD] business area alone, with annual costs of over $70 million...the total number of computers in DOD was estimated at 1046 with costs of about $443 million. On a rough approximation, DOD estimated that about half of these costs could be attributed to (1) systems design, (2) flowcharting, (3)

programming and coding, and (4) 'debugging.' With direct 'software' costs of about $35 million, which grew to over $200 million in the next five years, DOD obviously had an interest, as well as a position of stature, in the subject. Certainly, if such a project offered a hope of reducing 'software' costs, it would provide a strong motivation as well.

"Two other items were of concern to DOD at that time: compatibility and the interchange of computer programs; we might really consider them as two parts of the same basic problem. This problem was clearly evident in the supply and logistics area, which represented about 85 percent of DOD business-type applications in 1959." (p. 305).

20. According to the U.S. Commerce Department (1984), total U.S. software industry revenues amounted to $10.3 billion in 1982; in that year, the DoD Joint Service Task Force on Software Problems estimated that total Pentagon spending on software amounted to $5-6 billion (Department of Defense, 1982, p. 6). A portion of the revenues included in these estimates of defense-related software sales are excluded from the Commerce Department measure of industry revenues (and these revenue estimates may be low, judging from the comparison with data from the OECD cited in Chapter 1), but the orders of magnitude suggest that defense-related demand figured much more prominently within the software industry than in computer

hardware. Fisher (1978) notes that "At one time DoD was a major innovator and consumer of the most sophisticated computer hardware, but now it represents only a small fraction of the total market. In software, that unique position still remains: a significant fraction of the total software industry is devoted to DoD-related programs--and this is true in even larger proportion for the more advanced and demanding systems." (p. 24).

21. As the Defense Systems Management College study of "Mission Critical Computer Resources" (MCCR) noted, "In 1966 the FB-111 required an on-board computer memory of roughly 60,000 words but by 1988 the B-1B Bomber was approaching on-board computer memory requirements of about 2.5 million words. Current and future systems will greatly exceed these memory requirements with large scale software systems being the norm." (1990, p. 2-2).

22. The Congressional Office of Technology Assessment estimated that as of June 1987 the software development activities of the Strategic Defense Initiative Organization (SDIO) accounted for more than $275 million in expenditures (the estimate does not cover a single fiscal year's expenditures; instead, "...it shows money that at that time had been spent since the inception of the program, that was then under contract, or that was expected soon to be under

contract." (U.S. Congress, Office of Technology Assessment, 1988, p. 248).

23. Fisher (1978) estimated that embedded software accounted for 56%, dataprocessing software 19%, scientific software 5% and "other and indirect software costs" 20%, of total DoD software expenditures in 1973 (p. 25).

24. Joint Services Task Force (1982), p. 5. Other examples cited by the Task Force included modifications in the targeting accuracy of the Minuteman III nuclear missile and an emergency modification in the guidance systems of the British Sea Wolf antiaircraft missiles made to adapt them to the needs of the Falklands conflict in 1982.

25. Fisher (1978) noted that in 1978, "At least 200 models of computers are used in embedded computer systems at DoD. In many applications, the computers must be installed in configurations that are incompatible with general-purpose installations." (p. 25).

26. One other indication of this "divergence by design" is the fact that Ada does not incorporate object-oriented programming

design concepts or tools, which now are widely employed within the civilian software industry.

27. A partial listing of the major reports of the 1980s includes the following: the Joint Services Task Force Report on Software Problems (1982); the Software Engineering Institute Study Panel (1983); the Defense Science Board Task Force on Military Software (1987); the Department of Defense Software Master Plan (1990); and the Department of Defense Software Technology Strategy(1991). Among other things, this tabulation excludes the numerous studies of software issues in the Strategic Defense Initiative program. Pentagon studies of its software needs, of course, substantially predate the 1980s--the Joint Services Task Force report lists 26 previous studies in an appendix.

28. "Migration to commercial support spreads the maintenance costs for software technology across a much larger base than DoD. It is thus a major leverage factor for DoD software technology products. Even greater cost-effectiveness leverage can be obtained by stimulating existing commercial technology products to address DoD needs." (Department of Defense Software Technology Strategy, 1991, p. ES-23).

29. "The goal of the STARS program is to improve productivity while achieving greater system reliability and adaptability. Meeting this goal requires a very broad attack to improve the environment in which software is first developed and then supported. The DoD's 'Ada"' program provides an initial focus for the development of a common, sharable software base. The STARS program broadens the scope of attention to the entire environment in which software is conceived and evolved." (Druffel, Redwine, and Riddle, 1983, p. 10). The 1991 "Software Technology Strategy" report of the Defense Department noted that "The STARS prime contractors (IBM, Unisys, and Boeing) are developing these [software environment frameworks] in concert with their commercial counterparts (in-house for IBM and Unisys; DEC for Boeing) and a number of tool vendor subcontractors for computer-aided software engineering (CASE). The primary program objectives are that STARS products are commercially, supported, responsive to particular DoD needs (support of very large, embedded, real-time, and Ada software applications), and built using common open interfaces to facilitate CASE tool portability and interoperability. (1991, p. 3-4).

30. The Defense Department's January 1992 Software Technology Strategy called for increased reuse of code, in addition to improvements in software engineering and the use of more commercial software (Burgess 1992). Despite the substantial DoD investment in

software engineering, the commercial sector (especially in microcomputers) appears to be ahead of the defense sector in many of these techniques. For example, Microsoft is far ahead of traditional large-scale software houses (including IBM's now independent Federal Systems Division) in the use of such techniques as rapid prototyping (Smith and Cusumano 1993).

31. In 1991, DARPA funded a $22 million project on object-oriented data bases. The participants include Texas Instruments, the NIST, and several universities (Stix 1992). The April 1994 request for proposals for the ARPA-led Technology Reinvestment Program listed "Object Oriented Technology for Rapid Software Development and Delivery" as one if its areas of interest.

32. "The realization that software is more difficult to develop than hardware is one of the most important lessons of the Apollo program. So the choice of memory should be software driven, and designers should develop software needed for manned spaceflight near the Manned Spacecraft Center. The arrangement with MIT reduced overall quality and efficiency due to lack of communication. Also, more modularization of the software was needed." (Tomayko, 1988, p. 62).

33."The story of computers in manned mission control is largely the story of a close and mutually beneficial partnership between NASA and IBM. There are many instances of IBM support of the space program, but in no other case have the results been as directly applicable to its commercial product line. When Project Vanguard and later NASA approached IBM with the requirements for computers to do telemetry monitoring, trajectory calculations, and commanding, IBM found a market for its largest computers and a vehicle for developing ways of creating software to control multiple programs executing at once, capable of accepting and handling asynchronous data, and of running reliably in real time...

"The company maintained its lock on mission control contracts through Gemini, Apollo, and the Shuttle. At each point, some experienced personnel were transferred to other parts of the company to share lessons learned. Several individuals contributed to OS/360, the first multiprogramming system made commercial available by IBM. One became head of the personal computer division." (Tomayko, 1988, pp. 243-244).

34."IBM reacted to the increased complexity [of the Gemini program's data and mission control requirements] in several ways. Besides adding more manpower, the company enforced a strict set of software development standards. These standards were so successful that IBM adopted them companywide at a time when the key commercial

software systems that would carry the mainframe line of computers into the 1970s were under construction." (Tomayko, 1988, p. 252).

35. Sapolsky's comment on the changing role of the U.S. Office of Naval Research (1990) is relevant and prescient in this regard:

"National security rationalies are no longer very important in the support of basic research...But without the protection of national security rationales, science is vulnerable to political pressures in ways that undermine its integrity and productivity. When vital defense interests are not at stake, politicians wonder why their districts are not benefiting from the federal research largess much more than when they are. Less favored institutions and disciplines find the urge to employ pork barrel tactics impossible to resist. The network of elites that binds together the scientific community and provides its priorities cannot contain the desire for equity and opportunity that is so much a part of the political process. The Navy and the other armed services may not regret their reduced role in basic research, but science no doubt will." (p. 121).