Appendix F

Case Study: The Software Industry

CONTENTS

Appendix F
Case Study: The Software Industry

NOTE: This case study, along with those in Appendixes D and E, is presented in condensed form in chapter 9 of the main report, Holding the Edge.

INTRODUCTION

The word "software" means different things to different people. It encompasses everything from operating systems to home video games, missile guidance systems to database managers, file servers to compilers and translators. A more rigorous definition of software is: *the combination of data and computer instructions written in any of a variety of languages used to instruct or enable computer hardware to perform computational or control functions.* It includes computer programs, i.e., a series of instructions or statements in a form acceptable to a computer, designed to cause the computer to execute an operation or operations.

The software industry has advanced rapidly since it emerged in the early 1950s. At that time, computer programs were written in binary machine code specific to a particular mainframe computer. By 1955, assembly language, composed of abbreviated symbolic codes, replaced binary code as the computer instruction set. Soon after, the high-order languages FORTRAN (FORmula Translation) and COBOL (COmmon Business Oriented Language) were introduced, providing programmers with a more natural language interface to computers.[1] Since the late 1950s the computer industry has added a multiplicity of languages to the industry. By 1975, the Department of Defense alone used more than 450 computer languages and derivations of languages for its embedded systems.[2]

During this same period, advances in the hardware industry increased the performance of computers by six orders of magnitude and reduced the cost of computers by about the same amount.[3] Improvements in hardware have resulted in computer configurations ranging from personal computers (PCs) to mini-, micro-, and mainframe computers. Each type of computer is designed for a particular market and a different scale of applications. The variety of computers available at reasonable cost to the general public has stimulated a market for all types of software—the demand for some of which the software industry cannot currently meet.[4]

Increased demand, coupled with a global shortage of trained computer programmers and an exponential rise in the cost of developing and maintaining software, has created what some have called a "software crisis."[5] But the crisis presents ample opportunity to U.S. software fins, which have dominated the world software market since its inception, controlling 70 to 75 percent of the market share. The number of U.S.-based software firms has increased from 4,340 in 1982 to over 25,000 in 1987, with a corresponding increase in revenue from approximately $10 billion to $17.6 billion. The increase in the number of software firms can be explained in part by the introduction of the PC by IBM in 1981 and the resultant increase in demand for Software.[6]

Increases in both demand and revenues from software are expected to continue in the foreseeable future, but there are factors that may adversely affect

[1] U. S. Department of commerce, "A Competitive Assessment of 'I%c U.S. Software Industry," December 1984.

[2] Embedded computers are part of a larger (non-computer) system; automobiles, missiles, and microwave ovens all rely on embedded computers for their operation. Missile guidance systems are an example of embedded systems. Benjamin ElSon, "Software Update Aids Defense Program," *Aviation Week & Space Technology, Mar. 14, 1983,* p. 209. See also: Jeremy Tennenbaum, "The Military's Computing Crisis: The Search For a Solution" (New York, NY: Salomon Brothers, Inc., Sept. 22, 1987), p. 3.

[3] Frederick Brooks, Jr., "NO Silver Bullet: Essence and Accidents of Software Engineering," *Computer*, April 1987, pp. 10-19.

[4] John M[illegible], "Coming Up Short in Software," *Air Force Magazine*, February 1987, p. 64; and U.S. Department of Commerce, op. cit., footnote 1, p. 7.

[5] Dieter Ernst, *The Global Race In Microelectronics: Innovation and Corporate Strategies in a Period of Crisis* (Frankfurt: Campus Verlag, 1983).

[6] U.S. Department of Commerce, op. cit., footnote 1; and "Programming the Future," *The Economist,* Jan. 30, 1988, pp. 3-18.

the prosperity and health of the U.S. software industry in the world market. These include increasing competition from foreign companies, R&D that often focuses on short-term, application-specific software projects, foreign barriers to U.S. exports, inadequate intellectual property protection for U.S. developed software, and the failure of software technology to keep pace with advances in the hardware industry.

The health of this industry is vital to the nation's defense technology base because it profoundly affects DoD's ability to acquire and operate computer systems. In a very practical sense, software runs DoD. It controls communications systems among the Services, monitors force logistics, models scenarios of nuclear and conventional warfare, controls missile guidance systems, maintains accounting and payroll data provides office automation systems, and coordinates Command, Control, Communications and Intelligence (C^3I) operations.[7] The Office of Technology Assessment (OTA) is examining software as a dual use technology from the perspective of its contribution to the United States military. This case study examines the transfer of software technologies between the civilian-based and defense-related software industries, and the ability of the DoD to acquire and use the best available software technology.

This case study addresses three central questions. First, what is the current status and relative health of the United States software industry, both the defense-based and civilian-based industry? Second, what are the similarities and differences between the two sectors? Third, what procedural, institutional, technical or other barriers preclude the exchange of software technology between the defense and civilian sectors? Several policy options and problems generated from these three questions conclude this appendix.

GLOBAL SOFTWARE MARKETS AND THE HEALTH OF THE U.S. SOFTWARE INDUSTRY

Software is categorized in several ways-by its end-use application, by the size or scale of application, and by the degree to which it is customized. Industry and economic analysts use a variety of software classification schemes, and in some cases fail to distinguish between software as a service and as a product. It is therefore difficult to make generalizations about the software industry's nature, health, and future. Despite the lack of consistent economic data, the U.S. software industry clearly appears to be strong and competitive, in both the defense-based and civilian sectors.[8]

In 1981, U.S. software firms held 70 percent of the $10.3 billion international market for all types of software. In 1983, U.S. industry again controlled 70 percent of the world market, but the market had increased to $18.5 billion, putting the U.S. share at ***$13*** billion.[9] These figures are based on the international market for packaged software, integrated systems software, and custom-built software.[10] In 1983, approximately 60 percent of U.S. revenues came from packaged software, 25 percent from custom software, and the remainder from integrated systems software. In contrast, the other major software producing nations, Japan, France and the United Kingdom, receive most of their revenue from custom-built software, followed by integrated systems-software designed primarily for their respective domestic markets.[11]

---

[7] Software's significance to the defense technology base should not be underestimated: recent estimates show that DoD spends approximately $12 billion a year on its embedded software needs alone. Total software rests (including systems development and maintenance) are expected to consume 10 percent of the total DoD budget by 1990. See Jeremy Tennenbaum, op. cit., footnote 2; Jonathan Jacky, 'The Star Wars Defense Won't Compute," *Atlantic Monthly*, June 1985, pp. 18-30; U.S. Congress, Office of Technology Assessment, *SDI: Technology, Survivability, and Software*, OTA-ISC-353 (Washington, DC: U.S. Government Printing Office, June 1988), p. 225; and National Research Council, *The National Challenge in Computer Science and Technology* (Washington, DC: National Academy Press, 1988), p. 31.

[8] See for example, "National Academy Looks at Computing's Future," *Science, vol.* 241, Sept. 16, 1988, p. 1436; and U.S. Department of Commerce, op. cit., foomote 1.

[9] U.S. Department of Commerce, op. cit., footnote 1, pp. 32-34. In contrast to the Department of Commerce's figures, INPUT/ADAPSO repotted that in 1982, U.S. firms had revenues of $26.5 billion for the aggregate of: software products, data processing services, professional (consulting) services, and turnkey systems; *Software: An Emerging Industry* (part 9 of the series, "Information Computer and Communication Policy") (Paris: Organization for Economic Co-operation and Development, 1985), pp. 63-64.

[10] Packaged software is commercially developed and broadly marketed. Integrated systems software is a complete system that is sold and integrated with a specific hardware architecture. Custom software is developed to meet a user's specific needs.

[11] U.S. Department of Commerce, op. cit., foonote 1, p. 32.

By 1986, the U.S. share of the worldwide market for packaged software alone, including applications tools, generic solutions, and systems software, was $17.6 billion.[12] The worldwide market for packaged software is forecast to grow at a compound annual growth rate of 22 percent; and under this scenario, U.S. firms would reach revenues of $47.35 billion by 1991.[13] Various software industry estimates highlight the fact that no "hard" dollar figures are available since 1986, but most experts believe that regardless of the dollar amount, the United States still dominates the entire software industry.[14] The discrepancies in these estimates, forecasts, and reported revenues are partly attributable to an economic slump in the sales of hardware after forecasts were made, to variations in exchange rates, and to the classifications and methods used to report these figures.[15] They indicate the need for accurate measurement of the various types of software sold so that better analysis of the industry can be made.[16]

Although the U.S. software industry presently dominates the world market-both technically and economically-its continued superiority will depend on a number of complex factors. First, the industry faces a growing demand for all types of software-packaged, integrated systems, and custom built. Second, international competition in the industry is increasing as other nations-particularly Japan, France, the United Kingdom, Korea, and India—increase their software production capacity and penetrate the global software market. Third, U.S. software firms are increasingly forced to deal in an unfavorable international market where trade tariffs and national policies directly and indirectly restrict U.S. software exports to many foreign countries. And trade that does occur often fails to provide adequate intellectual property protection. Fourth, the gap between advances made in the hardware industry and those of the software industry continues to widen, making it difficult for the technologies of both industries to complement one another. Fifth, current software R&D activities relating to state-of-the-art technologies are insufficient. Finally, as the world market continues to grow, its composition will undoubtedly change, and the demand for new types of software may outpace that for current types, creating an advantage for newly established foreign companies. Each of these factors is addressed in a separate section below.

Software Supply and Demand

The ability of the U.S. industry to meet the overall demand for software depends on the various types of software that exist and current market trends associated with those types. Software is often categorized as either custom-built or packaged software. Custom-built software is developed according to a user's specific or unique requirements. An example is software developed to meet the requirements specified by a missile guidance system. Packaged software[17] consists of standardized software designed to be marketed widely. Examples of packaged software include operating systems, compilers, word processing systems, and database management systems. Prior to the existence of standardized operating systems and high-order languages, custom-built software dominated the U.S. software market. Since these developments, and as the cost of software has continued to increase relative to that of hardware, packaged software has increasingly dominated sales in the software market. Custom software accounted for almost one third of U.S. revenues in 1981. By 1983, the custom segment had fallen to about one fourth of total software sales with an annual growth rate of 16 percent. During that same period, packaged software sales grew at an annual rate of 40

12"Computer Industry Review & Forecast, 1982.1991 ," Special Report, International Data Corporation (IDC), October 1987, p. 109. This figure excludes custom-built software.

13Ibid.

14Information provided at the July 1988 Workshop on the Relationship Between Military & Civilian Software (hereinaftecalled OTA Software Workshop) suggested that the U.S. controlled art estimated 80 to 90 percent of the worldwide software market, for revenues of $30 billion, in 1988.

15Depending on the type or source of software, revenue/sales are reported under several Standard Industrial Classification (SIC) codes and other industry "de facto" classifications. For example, custom-built software is often accounted for in professional services, and integrated systems may be included partially in hardware sales and services. For further information, see: International Data Corporation, op. cit., footnote 12, pp. 108-109: and U.S. Department of Commerce, op. cit., footnote 1, pp. 3, 10-11.

16Source: OTA Software Workshop. Metrics of the industry and accurate classification of what constitutes software are needed not only to study industry trends, but to clarify how software is treated with respect to intellectual property protections, tax laws, accounting procedures and product liability laws. *Software: An Emerging industry, op.* cit., footnote 9, p. 11.

17Packaged software is also referred to as commercial-off-the-shelf (COTS) or Non-Developmental Item (NDI) software.

percent. This trend is expected to continue,[18] and is significant for the military which predominately acquires custom-built software for its applications.[19]

Packaged software can be further classified as **systems software,** including operating systems and systems support software; **applications tools,** including database managers, compilers, program development tools and environments; and **applications software,** software designed for a specific end-user problem, including generic banking, accounting, and office automation programs. Each of these segments of the packaged market is expected to grow in the future. Systems software, which currently makes up the largest share of revenues for U.S. firms, is expected to increase at the slowest rate and to decrease its market share. This reflects this market's symbiotic relation with the hardware industry. Systems software is typically developed for a particular size computer or specific hardware architecture, and recent fluctuations in the hardware market-particularly for mainframe computers-have negatively affected sales of these types of software.[20]

The ability to meet the growing demand for software, and of the United States to maintain its dominance of the softvare market, largely depends on the supply of computer programmers and the technology available to them. The United States cannot meet the demand for ail types of software with the present number of computer programmers. This shortage is not limited to the United States.[21] In 1985, the shortage of U.S. software professionals, including programmers, software engineers, and managers, was estimated at 50,000 to 100,000. There are many indications that this gap will continue to grow in the immediate future.[22]

The lack of qualified software developers maybe part of a larger shortfall in trained science and engineering professionals in the United States. Beyond any doubt, there is a serious shortage of rigorous software engineering programs at U.S. colleges and universities. The poor performance of American students in the sciences shows no immediate signs of reversal,[23] and while the number of students entering the computer science field seems to be increasing, demand outpaces estimates of future supply.[24] Finally, while there are signs that universities are adding more computer science and engineering courses to their curricula, an increasing

---

[18] U.S. Department of Commerce, op. cit., footnote 1, pp. 17-22. "

[19] According to GAO, as of 1983, between 95 and 98 percent of all software developed for U.S. Government agencies was custom built. Source: United States General Accounting Office, "Federal Agencies Could Save Time and Money With Better Computer Software Alternatives," GAO/AFMD-83-29, May 20, 1983, p. 4. It seems likely that the DoD has proportionately as least as much custom software as other Federal agencies, based on the DoD's numerous embedded systems, unique systems and languages, and security requirements. What has been acknowledged by many experta is that the DoD is increasingly using COTS software in its applications.

[20] International Data Corporation, op. cit., footnote 12, pp. 112-113. In 1986 systems software made up approximately 43 percent of the revenues received by U.S. firms in the packaged software market, applications tools made up 25 percent and applications software, 32 percent of that same market. By 1991, these shares are estimated to be 39 percent, 28 percent, and 33 percent respectively.

[21] Currently all industrial nations report a critical shortage of "software specialists," and most member nations of the Organization for Economic Cooperation and Development (OECD) identify this situation as the most important policy issue the software industry faces. *Software: An Emerging Industry*, op. cit., footnote *9, pp. 131-137. See also:* U.S. Department of commerce, op. cit., footnote 1, p. 72.

[22] Estimates that this *shortfall will* reach 1 million by 1990 in the U.S. alone are often cited, as are projections that demand for software professionals will exceed supply by 40 percent. *See* for example: John Morrocco, op. cit., footnote 4, p. 6% Paul J. McIlvaine, *"Software* Logistics: A Sleeping Giant," Concepts, Autumn 1982, p. 157; Parker Hodges, "The New Maturity of Computer Science," *Datamation*, Sept. 15, 1988, p. 40; Jeremy Tennenbaum, op. cit., footnote 2, p. 3; and *Software: An Emerging Industry,* op. cit., footnote 9, p. 131.

[23] See for example, "Science Achievement in Schools Called Distressingly Low," Science, Sept. 30, 1988, p. 1751, which indicates that the poor scientific understanding demonstrated by 9, 13, and 17 year-olds *poses* a serious threat to our national security. Also: American Electronics Association, "Engineering & Technical Education Program," September 1987.

[24] The supply of computer programmers is estimated to grow at a rate of 4 percent annually (Jeremy Tennenbaum, op. cit., footnote 2, p. 3), while the demand for computer programmers will grow at 70 percent and demand for computer analysts will grow at 76 percent (*Editorial Research Reports, Sept. 9,1988,* p. 446). See also: Parker Hodges, "The New Maturity of Computer Science,"Datumtzón, Sept. 15, 1988, pp. 37, 40; Office of the Under Secretary of Defense for Acquisition, *Report of the Defense Science Board Task Force on Military Software*, Washington, DC, September 1987, p. 38; U.S. Congress, Office of Technology Assessment, *Demographic Trends and the Scientific and Engineering Work Force—A Technical Memorandum,* (Springfield, VA: National Technical Information Service, December 1985); and American Electronics Association, op. cit., footnote 23.

percentage of students enrolling in these courses are foreign nationals.[25]

Software Quality

While efforts are underway to increase the number of individuals entering the software industry through improved education programs, these are long-term investments with payoffs not expected in the immediate future. Of more pressing concern is the quality of individuals entering the software engineering profession, and of those already in it. The complexity of applications, and the variety of hardware architectures that software is designed for, require scientific and engineering skills beyond those defined as computer programming.

Programming can be defined as the translation or written representation of a system design to a form interpretable by a computer. The actual translation or coding of statements in a high-order language requires minimum training to master. The difficulty in developing software is in the formulation of that design—the specification of data, data relationships, mathematical formulas and functions-in a rigorous and precise manner.[26] This process requires the software developer to conceptualize system complexity, interfaces to other systems, and future changes to the system. It is complicated by the fact there are no methods readily available that accurately represent the abstraction of all possible states that software can assume. The written computer program is a sequential translation that reflects only one such state. Graphical representations, such as flow charts or data flow diagrams, are similarly unable to capture all possible states, so that in both cases the design concept is retained only in the developer's mind.[27]

The difficulty in developing software is aggravated by what many consider to be the focus of computer science courses on software as a soft science, synonymous with coding, rather than an engineering science.[28] As a result many new programmers are unskilled in large-scale systems development and in the maintenance of such systems.[29] They may have limited experience working as part of a project team, but do not understand the engineering and design principles necessary to build real-world systems. Because the capabilities of computer programmers and software engineers directly affect the productivity and health of the industry as a whole, rigorous educational and training programs are a critical factor in the health of the software industry.

The software development process can be improved through the use of formalized and automated engineering techniques that support the iterative building and testing of software prototype systems, allow for the reuse of software components, and accommodate the complexity of software systems. Many software development methods and practices used today are primitive when compared to sophisticated software engineering techniques. It is not uncommon to find programmers using practices 10 years behind today's most advanced technology, due to inertia and the incompatibility of existing systems with these techniques. The result is that many software tools and concepts commonly used lag far behind those of the hardware which that software controls. Software utilities and Computer Aided

---

[25]Approximately 39 percent of the computer science/engineering PhDs awarded in the United States in 1987 were granted to foreign nationals. David Gries and Dorothy Marsh, "The 1986-1987 Taulbee Survey," Special Report, *IEEE Computer*, March 1985, p. 53. See also: Parker Hodges, op. cit., footnote 2. It has been noted that some non-U.S. citizens who "partake of the educational opportunity in the U.S." do in fact stay in the United States, but that more often these students take U.S. technology back to their homeland. This is because of increased opportunity in their homeland or because of national policy that mandates the return of these students upon the completion of their education (as is the policy in Singapore) or as part of a contract with a corporation which has funded the training (OTA Software Workshop). See also: American Electronics Association, op. cit., footnote 23, p. 12; and Robert Park, "Restricting Information: A Dangerous Game," *Issues in Science and Technology*, Fall 1988, pp. 62-67.

[26]It is generally agreed that almost anyone can be trained to write computer programs in a high-order language (HOL), but the entire software development process is a rigorous one and requires systems analysis, design, and engineering skills that are not readily mastered.

[27]For further discussion of the difficulty in developing software, see: Frederick Brooks, Jr., op. cit., footnote 3.

[28]Structured programming refers to the decomposition of a computer program design into manageable, modular or functional pieces and the subsequent "coding" of these pieces. Software engineering, while encompassing this process, emphasizes the entire systems lifecycle development process. It includes the technologies and methodologies for conducting requirements analysis, risk analysis, quality assurance/software testing, configuration management, implementation, and future modifications of software. For further information, see *Software: An Emerging Industry*, op. cit., footnote 9, pp. 11-12, 30-39; and U.S. Congress, Office of Technology Assessment, *Information Technology R&D: Critical Trends and Issues* (Springfield, VA: National Technical Information Service, 1985), pp. 79-85.

[29]The problems associated with developing software programs of increasing size are detailed in Fred Brooks, *The Mythical Man-Month* (Reading, MA: Addison-Wesley, 1975).

Software Engineering (CASE) techniques that are available are employed erratically in the industry. These factors contribute to the impression that software state-of-the-art is still art, not science.[30]

Appropriate and leading-edge technology is critical to the development of correct and maintainable software. Current practices and conditions—the failure to recognize software engineering as a scientific discipline and the lack of trained software engineers—am largely responsible for the growing cost of maintaining operational software systems. Maintenance, which encompasses the modification of software both to correct errors and to incorporate changes or enhancements, has become the major cost in any software system. Maintenance costs consume between 50 and 80 percent of all software budgets. Present estimates indicate that in fiscal year 1990, DoD will spend 80 percent of its software budget ($16 billion) on maintenance. Industry-wide maintenance costs are estimated to exceed those for development by a factor of 50.[31] While no software can be expected to be error free, the use of engineering techniques, system prototypes, modular system development, standard languages, and CASE tools can minimize computer "bugs" and improve productivity. Additionally, these practices support the development of portable and upward compatible systems that accommodate future enhancements and modifications.[32]

DoD has responded to the software crisis in two ways. First, the Department mandated the use of a standard language, Ada, which supports the use of modem software engineering practices and which is designed to replace the multiplicity of computer languages used in the DoD for mission-critical systems. Second, DoD has stated a preference for the use of commercial-off-the-shelf software wherever possible.[33]

Foreign Competition

Today, approximately 40 percent of the packaged software revenues earned by U.S. firms come from outside the United States.[34] This share is threatened by the software industries of Japan, France, the United Kingdom, Korea, India, Taiwan, and Singapore.[35] Japan is the strongest competitor primarily because of its strong hardware industry and propensity to take advantage of standardized technologies and develop marketable products from them.[36] The strength of the Japanese, and to some degree Singapore, India, and Taiwan, is in their ability to close large portions of the world market to the United States and simultaneously penetrate the U.S. market with systems software created with U. S.-developed technology. The quality of these products is equal to those of the U.S. firms, and can partly be attributed to the facts that many foreign engineers are trained in the United States and that a number of U.S. firms have established development facilities overseas. However, the quality of other types of software developed in these nations, especially

applications software, suffers in comparison to that developed in the United States[37]

A comparison of the U.S. and Japanese industries shows that, while the level of software technology in both countries is similar, Japanese firms establish more disciplined software engineering environments conducive to the development and use of software tools. Japanese firms make greater investments in the area of basic technology and distribute this capitalization within the entire firm, rather than localizing it to a particular software project as is typically done in the United States. Additionally, the Japanese incorporate experiences and lessons learned into their future projects.[38] The Japanese are, in fact, turning programming into an applied science as demonstrated by "software factories" that reuse approximately 30 percent of previously developed software, have an error rate one-tenth that of their U.S. counterparts, and have the potential to produce lower cost and higher quality software.[39] The result of these efforts is programmer productivity figures that greatly exceed those in the United States. However, many experts note that at least part of the discrepancy between Japanese and U.S. productivity and error rates can be attributed to the fact that much Japanese software production focuses on programming from extant design. Further, these figures tend to be balanced by more efficient project management practices in the United States.[40]

The third major competitor in the worldwide software market, France, receives a considerable portion of its revenues outside its own borders. This contrasts with Japan's larger, but almost exclusively domestic, market. As a result, France was second only to the United States in worldwide software sales in 1982. The strength of the French industry is partly a result of national policy and partly the result of its growing internal software needs.[41]

Competition from other foreign nations is partly the result of industry standards. The development of standards is seen as a mixed blessing in the software industry. Although a lack of standards spawns innovation and creativity, it can also create excessive numbers of incompatible systems that inhibit rapid development of the technology. It is generally agreed that standards are needed and appropriate for systems interfaces, computer languages and protocols, and they are useful in codifying existing practices. But this also makes it easier for foreign vendors to compete effectively in the software market by taking advantage of technology developed by others.

International Software Protection and Trade Policies

As U.S. software firms exploit the world market, they become increasingly subject to intellectual property violations and infringements by foreign vendors. U.S. intellectual property protections (copyrights, trademarks, trade secrets, and proprietary data) are currently insufficient to protect U.S. interests in foreign nations where penalties for intellectual property infringement are less than the potential profits to be made from such infringement. Foremost among the violators are lesser developed and newly industrialized countries-Taiwan, Korea, and Brazil-which have little to lose and much to gain by not honoring U.S. regulations. Japan is also cited frequently for violations. The International Trade Commission surveyed over 400 U.S. firms in 1986, and estimated that U.S. computer hardware and software firms lost $4.1 billion due to inadequacies in intellectual property protection. This figure includes loss of exports and domestic sales to foreign infringing goods and counterfeit products, unrecovered research costs, increased product liability costs, reduced profit margins, damage to corpora-

[37]Ibid., pp. 163, 168.

[38]Marvin Zelkowitz et al., op. cit., footnote 30.

[39]Ware Myers, op. cit., footnote 30, p. 89; see also: U.S. Congress, Office of Technology Assessment, op. cit., footnote 28, p. 85; U.S. Congress, Office of Technology Assessment, *International Competition in Services: Banking, Building, Software, Know-How . . .*, op. cit., footnote 35, p. 164; and U.S. Department of Commerce, op. cit., footnote 1, p. 11.

[40]U.S. Department of Commerce, op. cit., footnote 1, p. 11; and U.S. Congress, Office of Technology Assessment, *International Competition in Services: Banking, Building, Software, Know-How . . .*, op. cit., footnote 35, p. 164.

[41]U.S. Department of Commerce, op. cit., footnote 1, pp. 38-40; and U.S. Congress, Office of Technology Assessment, *International Competition in Services: Banking, Building, Software, Know-How . . .*, op. cit., footnote 35.

tion trademark or reputation, and lost employment opportunities. [42]

These losses translate into decreased incentive for affected firms to invest in new technologies and innovative research and development activities.[43] Three conditions appear to encourage this situation. First, the technology and resources required to produce counterfeits or imitations of legitimate software products are readily available and relatively inexpensive. Second, consumers remain indifferent to or unable to detect differences between legitimate and infringing products. And third, the cost of genuine innovation remains higher than that of imitation. As long as these conditions prevail, the problem of inadequate intellectual property protection for software will remain.[44]

Additional economic loss is attributed to restrictive trade policies that serve to foster native software industries at the expense of U.S. firms. Import quotas, discriminatory taxes, local ownership requirements, embargoes, trade tariffs, and preferential treatment for locally produced goods are among the common policies which discourage or preclude U.S. firms from seeking business in many foreign nations. These practices are most pronounced in Brazil, India, Mexico, and Korea.[45]

Hardware's Impact

A major portion of the software industry is intimately related to the hardware industry. This is particularly true for systems software and, more recently, packaged software geared to the PC market. Since the preponderance of computer manufacturers are U.S. based, this symbiotic relation has traditionally benefited the U.S. software industry. While efforts are underway to diminish this strong tie to the hardware industry-for example, 0S/2, UNIX, and MS-DOS-this relationship will remain as long as the demand for integrated systems software and software development environments designed for particular hardware architectures continues. [46]

The increasing complexity of software systems, and the inability of software technologies to keep pace with innovations in the hardware industry, is of great concern.[47] The gap between the hardware and software industries can be seen in the exponential rise in software costs relative to hardware costs, the low productivity growth rates of programmers,[48] the increasing incompatibility of software systems, and the high costs associated with integrating or retrofitting existing software for new distributed architectures.

The United States seems unable to take full advantage of many of the advanced hardware and software technologies it has developed, principally because of its large embedded and heterogeneous software base. The problem of technology insertion is exacerbated by inadequate provisions in the software for its maintenance or inevitable post-delivery modification. Many existing military and civilian software systems are old by software state-of-the-art standards, but young with respect to their life expectancy of 5 to 20 or more years. Their longevity implies that the potential to use many advanced or new technologies is limited to software "maintenance" or modifications. Too often, such changes are not considered in the design process—functionality and data structures are not isolated and there is no system modularity to accommodate

[42]United States International Trade Commission, "Foreign Protection of Intellectual Property Rights and the Effect on U.S. Industry and Trade," USITC Publication 2065, Washington, DC, February 1988. Violations in Taiwan and Brazil accounted for over $1.05 billion of all lost revenues reported by U.S. computer firms to the ITC. In 1986, the Software Publishing Association reported that U.S. software manufacturers lost $1 billion in sales due to piracy. See Anne Branscomb, "Who Owns Creativity? Property Rights in the Information Age," *Technology Review*, May/June 1988, p. 42.

[43]United States International Trade Commission, op. cit., footnote 42, p. 4-1.

[44]Ibid., p. 4-7. Most respondents to the survey indicated their belief that the situation regarding intellectual property protections has deteriorated in the past 15 years, and expressed little hope of an improved environment in the near term. pp. 5-1 - 5-3.

[45]Ibid., pp. 3-12, 3-13, G-14.

[46]At this time the only major competitor to U.S mini-and mainframe computer manufacturers' 80-85 percent control of the computer equipment market, and therefore much of the related system software for these computers, is Japan. U.S. Congress, Office of Technology Assessment, *International Competition in Services: Banking, Building, Software, Know-How . . .*, op. cit., footnote 35, pp. 168-169. See also: Ware Myers, op. cit., footnote 30, p. 85.

[47]Edward Joyce, op. cit., footnote 30; John Morrocco, op. cit., footnote 4, pp.64-68; and "Summer Study 1985, A Report to the Director, SDIO," op. cit., footnote 31, p. 48.

[48]In 1983, U.S. software productivity grew at 4 percent while demand for software grew at 12 percent. Source: Jeremy Tennenbaum, op. cit., footnote 2, p. 4.

change. As a result, maintenance becomes a costly and time-consuming proposition. Finally, many new technologies and methodologies are incompatible with the computer language or dialect used in the original software." As a result of these factors and the United States' commitment to its software base, the United States is it at a relative disadvantage to those nations just entering the computer industry that have little or no historical computer base.[50]

R&D Investment

The present software crisis indicates the need for reinvestment and capitalization in the U.S. software industry that fosters R&D and technological growth and provides the capacity to exploit advances made in the industry. It is estimated that Japan spends approximately two-thirds of its R&D budget on process innovation, while the United States spends only one-third of its R&D monies on the same activities .51

Currently, the U.S. Government funds several software-related research efforts. The Software Engineering Institute (SEI) is a Federal Research Center located at Carnegie-Mellon University. It is responsible for numerous R&D projects relating to software productivity, reuse, and education. The objectives of DoD's Software Technology for Adaptable, Reliable Systems (STARS) include identifying possible technical solutions, methodologies, and tools that can be used to build reliable and cost-effective defense-based software. Without continued commitment to these programs, and further funding to support research and development in the areas of software engineering, development environments, distributed systems, and software metrics that record these efforts, it is likely that improvements in software productivity, cost, and reliability will be realized and put into common practice more slowly than necessary.[52]

CONVERGENCE/DIVERGENCE OF CIVILIAN AND MILITARY SOFTWARE TECHNOLOGY

The software industry is increasingly divided into two groups, one dedicated to military interests and another that supplies the commercial world.[53] These two sectors have always been present, and exchange between the two was assumed to be the norm, not the exception. But these groups seem increasingly to be diverging, a trend that is contributing to a weakening of the U.S. software technology base. As a major consumer of software and software services, the DoD has exerted, and will continue to exert, much influence over developments in the industry. Therefore, a strong software industry, one where the technology and research base is not divided between military and civilian environments, is in the interest of both communities.[54]

Convergence of Civilian and Military Software Technology

As in many other industries, the underlying software technologies are highly similar in both the military and civilian sectors, and divergence only becomes noticeable in the detailed requirements for specialized applications. Convergence between civilian and military software industries is most noticeable in the small-scale applications and systems software areas. Both sectors use packaged/COTS software for the majority of their small-scale software applications. These include PC-based pro-

[49]See: Frederick Brooks, Jr., op. cit., footnote 3, p. 12; Office of the Under Secretary of Defense for Acquisition, op. cit., footnote 24, pp. 8-12; and "Suggestions for DoD Management of Computer Software," op. cit., footnote 30, pp. 64-67.

[50]Based on discussions at OTA's Software Workshop, held July 25, 1988. See also: *Software an Emerging Industry*, op. cit., footnote 9, pp. 21-63; and *Discriminate Deterrence*, op. cit., footnote 35.

[51]The importance of focusing R&D in innovative and leading-edge technologies as opposed to risk-free, conservative, market-based technologies is stressed in *Discriminate Deterrence*, op. cit., footnote 35; and "Picking Up the Pace—the Commercial Challenge to American Innovation," Council on Competitiveness, 1988. These reports and others, as well as comments made during OTA's Software Workshop, indicate the failings of the United States' short-term view of R&D activities and its emphasis on DoD weapons-based activities as opposed to project-independent, long-term, and innovative R&D.

[52]The importance of R&D is noted in, for example: U.S. Congress, Office of Technology Assessment, *International Competition in Services: Banking, Building, Software, Know-How . . .*, op. cit., footnote 35; and U.S. Congress, Office of Technology Assessment, op. cit., footnote 28.

[53]While the word "commercial" or "civilian" seems to encompass any for-profit company, in this analysis these terms exclude those firms that exclusively contract with the government (DoD) or those divisions of a firm that are explicitly established to do business with DoD; e.g., IBM Federal Systems must be considered part of the "military" industry since its management structure is distinct from that of other IBM activities. The management characteristics of government contractors will be addressed further in the later section on "Barriers."

[54]John Morrocco, op. cit., footnote 4, p. 64.

grams and office automation products. Packaged systems software, such as operating systems, compilers, and systems utilities, are used to the same degree in both environments as well. The basic requirements for these particular types of software are similar in both sectors, and there is little need for customization of these products. More importantly, the availability and cost of these types of packaged software products make them readily accessible and attractive to both military and civilian users.[55]

Convergence in the industry's two sectors is also evident in their acceptance of CASE tools and modem software engineering methodologies. Unfortunately, this convergence is not always at the state of the art. Experts from both sectors of the industry cite examples of the use of modern engineering technologies that increase productivity and performance; but they are quick to acknowledge that at least as many software projects use little or no advanced technology.[56] The unpredictable and varied use of modem software engineering techniques and tools throughout the software lifecycle[57] is not localized by organization. Discrepancies are found within the DoD Services and agencies, within civilian firms, and within software projects of both sectors. A probable explanation for the industry-wide discrepancy is in the relative age of the system being analyzed. New starts and recently developed systems are more likely to exploit new technologies; they will be implemented in high-order languages, and modem engineering techniques will be brought to bear in their design and development.[58]

Divergence of Civilian and Military Software Technology

In general, the military and civilian software industries have access to, and use, the same technology. But they diverge in the ways they acquire software and, in particular, at the point where they sponsor large-scale applications that require custom-built software.

Similar applications for software are not limited to the PC-based or systems software previously mentioned. Analogous applications of large-scale software systems can be found in both sectors as well and include software developed for avionics, telecommunications, and embedded systems. But while the applications are similar, military and civilian environments place different, sometimes opposing, requirements on the software that controls these systems.[59] This is particularly true for large-scale, *mission-critical* military applications.[60]

Different requirements, as well as differences in scale, create two distinct software industries in the large-scale applications area. The industry divergence is illustrated in avionics systems software, where military requirements for high performance avionics are exchanged for high survivability and safety in civilian avionics.[61] The significance each sector attaches to software requirements, and whether they become rigid specifications or economic trade-offs, partially explains why there is

[55]Source: OTA Software Workshop, July 25, 1988. Each sector is as likely to "tweak" COTS systems software (e.g., compilers) for its own performance, efficiency, or other requirements and thus end up with unique software; but ultimately the source is the same.

[56]The emphasis on modern software engineering is most apparent in both the military and civilian aerospace industries. Source: OTA Software Workshop. For further information on the disparate use of these practices, see Marvin Zelkowitz et al., op. cit., footnote 30, pp. 57-66, which indicates that software practices typically used in 1983 were 10 years behind the research of that time.

[57]Software lifecycle refers to all stages of the software development process, from its conceptualization to its destruction. It normally includes requirements analysis, design analysis, implementation or coding, testing, deployment, and post-deployment (maintenance).

[58]While neither sector seems to lead in the use or implementation of modern software engineering techniques and tools, software industry analysts agree that the DoD, through its mandate for Ada and software initiatives, seems more verbally supportive and encouraging of efforts to use state-of-the-art technologies.

[59]Source: OTA Software Workshop. Most analysts agree that there is an analogous distribution of software applications in both sectors, and that the basic direction of software technologies is similar but parallel. Basically, the military and civilian sectors have the same problems in two different domains.

[60]Mission critical software is that which controls weapons, command and control, intelligence and similar systems.

[61]Source: OTA Software Workshop. The *Report of the Defense Science Board Task Force on Military Software* (op. cit., footnote 24, pp. 6-7) characterizes military software as "fundamentally like advanced civilian software, only more so." This refers to the fact that the requirements of military mission critical applications stress the state of the software art more severely than do similar civilian applications.

little transfer of software between them at the embedded and large-scale application levels.[62]

In contrast to civilian industry, military requirements for custom-built and embedded software tend to be very rigid. Once documented and approved in the design stage, the specified requirements govern the subsequent development of the software, regardless of their criticality to the system. Any such change typically requires a System Development Notification and contract modification that delay development. In addition to user-specified requirements, military software systems must address the maintainability, survivability, security, availability, reliability and interoperability[63] aspects of software.[64] These requirements are usually specified in absolute terms, not all of which maybe necessary for a particular military system. But they are more easily copied from previous software contracts than tailored for the new system.[65] The need for specific performance and operational characteristics is evident in many DoD mission critical-systems. It is necessary to require near-loo" percent reliability for a missile guidance system and desirable to require multi-level security in a networked defense communications system. But when these requirements are unnecessarily transferred to other military systems, the cost of development increases and the ability to use analogous civilian applications or commercially developed software decreases.[66]

Many of the requirements identified as unique to military application—e. g., security, data encryption, interoperability, survivability, and reliability—are appropriate in banking, insurance, commercial flight control, and other civilian applications. Indeed many of the characteristics implemented for military purposes could be transferred to civilian applications.[67] But while these requirements are desirable and appropriate in civilian applications, their implementation would be based on economic and risk analysis. The bottom line in the civilian sector is economic. If the cost of implementing a requirement exceeds the expected return for that implementation, then the requirement is, in most cases, deleted or deferred. This analysis and design-to-cost approach rarely occurs in military software acquisitions, although similar accommodations will be more likely in the future if the cost of military software continues to escalate. In contrast to the civilian methods, military software is designed to a set of approved requirements that seeks to minimize cost and risk; often these requirements fail to distinguish between the user's needs and wants.

The requirement for custom-built software exists equally in both sectors, but custom-built software appears to be used more often in DoD applications. The General Accounting Office (GAO) reported in 1983 that 95 to 98 percent of applications software used by the government was custom-built.[68] There are indications that the military is increasingly using commercially developed software in its systems; nevertheless, the majority of mission-critical and embedded systems software is still custom-built. One report estimates that "custom development will exceed packaged software sales in the Federal segment, in contrast to the mass market, where COTS software products will exhibit more rapid growth."[69] The trend to use more COTS products acknowledges that the disadvantages of using commercially available software-not receiving the customized software to meet unique requirements—are clearly outweighed by the direct and indirect benefits of using such software. These include cost savings, improved documentation and operational support, and increased availability. As the relative

[62]There are instances of interaction and technology exchange between the two sectors in these applications areas, most notably in the areas of avionics and communications. But because this is not the norm, the majority of this software is custom-built exclusively for one sector or the other. The limited exchange is largely the result of government regulations and auditing procedures required of government-funded development efforts. This problem will be addressed in the "Barriers" section of this appendix.

[63]Interoperability is the degree to which systems can exchange data with each other without affecting the operation of either system.

[64]The development of defense system software, unless otherwise specified in the contract's Statement of Work (SOW), requires contractors to develop a Software Requirements Specification (SRS) which includes these and other standard requirements and enumerates to what extent they will be met by the software. See DoD-STD 2167A and Data Item Description (DID) DI-MCCR-80025 for required elements and their format.

[65]Source: OTA Software Workshop. See also "Suggestions for DoD Management of Computer Software," op. cit., footnote 30, p. 69. The practice of using existing contracts as the basis for a new procurement is understandable, given the number of Military Standards, Data Item Descriptions, and documents required by defense acquisition regulations.

[66]It is estimated that achieving the close-to-perfect levels of performance required by some systems raises the development cost 30 to 50 percent. Although this cost factor applies to performance at the margin for weapons systems, there is a direct correlation to software systems—ensuring near 100 percent reliability for a software system affects development costs similarly. Jacques Gansler, "Integrating Civilian and Military Industry," *Issues in Science and Technology*, Fall 1988, p. 70.

[67]Source: OTA Software Workshop.

[68]United States General Accounting Office, op. cit., footnote 19.

[69]Darryl Taft, "Fed's Software Buys to Hit $3 Billion in '92," *Government Computer News*, Aug. 29, 1988, p. 41.

cost of acquiring and developing custom software continues to increase, so does the trend to use COTS products.[70]

An approach intermediate between COTS and custom-built software is the customization of commercially developed software or reuse of existing software. As the technologies to support reuse mature, one would expect both sectors to adopt this practice and incorporate previously developed software as components of larger, integrated systems .71 The degree to which reuse is accomplished by either sector is not known, but it is an area of potential convergence. Whether economic reality in the civilian sector is likely to encourage this practice more than Directives and mandates issued by DoD may depend on the current DoD requirements and procedures that discourage contractors from adopting this practice. According to many experts, there are currently few economic incentives, particularly in "cost plus" contracts used by the DoD, for contractors to reuse existing software; building software is perceived to be more profitable than reusing software.[72]

Divergent Acquisition Procedures and Lifecycle Model

Much divergence between civilian and military software is related to the acquisition process. It is evident in the way in which software requirements are specified, in the design and development of software, and throughout the entire software lifecycle. This divergence is magnified in the areas of special applications and large-scale systems software.

The analysis and writing of system requirements based on a user's needs is the most critical and difficult aspect of developing software.[73] Once established and approved, requirements directly influence the entire design and development of software. It is therefore essential that software requirements accurately reflect the needs of the user; that they do not place impossible performance, interoperability, or similar demands on the software; and are not so rigid that they preclude inevitable modifications to the software. The optimal way to accomplish this crucial task is to develop software requirements iteratively .74 Success ultimately depends on having a flexible vehicle that allows for iterative development, not only of requirements, but of the entire software lifecycle.

The mechanism used by the military is DoD Standard 2167A, which establishes the "requirements to be applied during the acquisition, development, or support of software systems." DoD-STD 2167A is designed to provide flexibility in the software development process, and at the same time provide the DoD with a mechanism to monitor that process.[75] Its objectives have not been fully realized because many government procurement officers still follow the older "waterfall" lifecycle model of software development exemplified in DoD-STD 2167.[76]

The waterfall lifecycle identifies distinct stages during the life of software that are associated with requirements analysis and definition, system design, system implementation or coding, systems testing, and deployment (including maintenance). Based largely on weapons acquisition, the military interprets this model to describe a sequential software development process, where each stage of development naturally leads to the next. Each phase is documented and is normally accompanied by a government review. Once the system requirements

The analysis and writing of system requirements based on a user's needs is the most critical and documented and is normally accompanied by a government review. Once the system requirements

[70]Frederick Brooks, Jr., op. cit., footnote 3, pp. 16-17.

[71]Software reuse includes the reuse of software designs, data, algorithms, software modules or entire programs. Technologies are needed to identify and develop appropriate reuse components, catalog these items, and retrieve them. Additionally, issues such as Quality Assurance testing, product liability, and intellectual property protection for reuse components must be resolved. See: Edward Joyce, op. cit., footnote 30; Tom Durek and Fred Van Horne, "Systems Software Development: Building Canonical Libraries," *Signal*, April 1988, pp. 89-93; and Frederick Brooks, Jr., op. cit., footnote 3.

[72]Source: OTA Software Workshop.

[73]Frederick Brooks, Jr., op. cit., footnote 3, p. 17.

[74]Ibid., pp. 17-18; and "Suggestions for DoD Management of Computer Software," op. cit., footnote 30, p. 66.

[75]*Defense System Software Development*, Military Standard, DoD-STD 2167A, Feb. 29, 1988. 2167A encourages the use of an appropriate

are specified and system design is complete, it is assumed that the implementation can and will automatically fall out from that design. In reality, the software development process is evolutionary and requires an iterative approach.[77]

While DoD-STD 2167A was designed in part to correct the waterfall lifecycle bias currently used, it continues to emphasize a document-driven, specify-then-build approach to software development.[78] The procedures set forth by DoD-STD 2167A and corresponding documents are designed to ensure that DoD gets the highest-quality software at the best price. But the system has not improved the quality or timeliness, or decreased the cost, of military software. Instead it remains a major part of the problem. The military's approach is based largely on competitive procurements that necessitate establishing requirements as early as possible in the lifecycle. The process backfires, however, once bids are awarded; many requirements turn out to be impractical, beyond the scope of current technology, or simply unneeded. The inevitable result is software that is delivered late, at higher cost, and with less functionality than planned.

DoD-STD 2167A attempts to avoid the cascade effect of this approach by allowing for all lifecycle stages to occur iteratively. But the standard directly or indirectly requires that developers comply with numerous other DoD and Military Standards, Directives, and Data Item Descriptions at each major development stage, milestone, or prior to a major revision in order to provide government oversight of the entire process.[79] These procedures perpetuate the inflexibility and bureaucracy that DoD-STD 2167 originated. The acquisition process used by DoD illustrates the government's propensity to use process specifications and standards that dictate how-to-design and how-to-manage, rather than performance specifications and standards that focus on desired results.[80] This approach contrasts with the civilian sector's tendency to negotiate for a final product and design-to-cost, and precludes the use of innovative or unproven techniques and methodologies by DoD contractors.[81]

Ada

A more recent divergence in the two industries relates to the military's mandated use of a single high-order language, Ada, in its mission-critical software systems. DoD's sponsorship for Ada began in 1974 when the "software crisis" was acknowledged to have potentially serious consequences for the military's ability to maintain and operate its many computer systems.[82] In 1983, Ada was approved as an American National Standards Institute (ANSI) and Military (MIL-STD 1815A) standard. By 1987, Ada was approved as an International Standards Organization (1SO) standard.

The DoD Directive that Ada shall be the single high-order language used in command and control, intelligence, and weapons systems has no counterpart in the commercial environment. With the exception of civilian avionics systems, Ada is not widely used in U.S. commercial applications. Instead, civilian-based software continues to be implemented in the language thought best for that application—whether it be COBOL, Assembly, a

[77] The benefits of an iterative approach and prototyping of systems development, are described in: Frederick Brooks, Jr.. op. cit.. footnote 3; Office of the under Secretary of Defense for Acquisition, op. cit., footnote 24, pp. 11, *33-35;* U.S. Department of Commerce, National Bureau of Standards, '*Application Software Prototyping and Fourth Generation Languages," NBS Special Publication 500-148, May 1987; John Morrocco, op. cit., footnote 4; Mark Gerhardt, "The Language of Abstraction," *Aerospace America,* July 1988, pp. 32-34; and U.S. Congress, Office of Technology Assessment, op. cit., footnote 28, The ill-effects of the waterfall lifecycle and DoD-STD 2167 are addressed further in the following section on barriers between the military and civilian sectors.

[78] Office of the Under Secretary of Defense for Acquisition, op. cit., footnote 24, P. 33.

[79] DoD-STD 2167A, op. cit., footnote 75, see pp. 1-4, 35-36.

[80] Donald Firesmith, "Should the DoD Mandate a Standard Software Development Process," Proceedings of Joint Ada Conference on Ada Technology and Washington Ada Symposium, March 1987, pp. 159-167.

[81] Source: OTA Software Workshop.

[82] A 1983 memorandum from the Under Secretary of Defense for Research and Engineering recommmended that Adabe the single high-order language used in all DoD mission-critical computer systems; but this recommendation was not implemented until 1987 in DoD Directive 3405.1, which states that Ada shall be the *'single, common, computer programming language" used in command and control, intelligence, and weapons systems. Policy regarding Ada is also provided in DoD Directive 3405.2, which mandates the use of Ada in all weapons-related computer systems.

Fourth Generation Language,[83] or any other computer language.[84] As new DoD computer systems are developed, the convergence of new software technologies and the ability to transfer software between the two sectors will depend a great deal on several factors: first the civilian sector's acceptance of, and demonstrated use of, Ada; second, DoD's willingness to grant waivers to its Ada mandate; and finally, the military's acceptance or ability to incorporate commercially developed, non-Ada software in its computer systems. The barriers potentially introduced by Ada will be examined further in the following section.

BARRIERS TO MILITARY ACCESS TO CIVILIAN SOFTWARE SECTOR AND VICE VERSA

Despite similarities in technologies available to the civilian and military software sectors, it is apparent that there is a growing divergence between them. Such differences, primarily in their respective acquisition strategies, obstruct the exchange of software technologies and applications. This continuing divergence not only damages the U.S. software industry, but also erodes the defense technology base. Previous studies, reports, and directives have identified the importance of technological exchange between the commercial-based and military-based software industries, and the need for DoD to adopt a more commercial-like acquisition process. But these reports, prepared by the Defense Science Board and others, have had little impact on the systemic problems identified to date. The persistent barriers to the transfer of technology, methodologies, and products between military and civilian interests are identified below.

Acquisition Regulations

In 1987, a Defense Science Board Task Force reported that the "major problems with military software development are not technical problems, but management problems."[85] This finding was revised during a follow-up workshop to state that while both technical and management problems are evident in military software development, the latter are more significant. These management problems relate to the manner in which the DoD procures software, and they represent the major barriers to the exchange of software technology between civilian and military sectors.

According to industry representatives, the principal barrier to exchange of software technology between the civilian and military sectors is the bureaucracy and administration overhead associated with DoD acquisition procedures. Requirements regarding the procurement, design, and development of DoD software are enumerated in DoD-STD 2167A, which provides "the basis for government insight to a contractor's software development, testing, and evaluation efforts."[86] DoD-STD 2167A does not profess to follow a particular software lifecycle model and does not require a particular software development methodology. Yet, as a review mechanism, it unnecessarily burdens contractors with the many Standards, Directives, Data Item Descriptions, and Federal Acquisition Regulations that it directly or indirectly requires.[87]

DoD has defined eight major activities for software development, each of which requires a formal review or audit to be conducted or supported by the contractor. Additionally, the contractor must document his plan for completing all activities and DoD must approve this plan before any development

[83] Fourth-generation languages arc application-specific languages or program generator% not geaeral-purpose or high-order languages. They include database languages, spreadsheets, natural query languages, and any language designed to be used in a limited problem domain. Office of the Under Secretary of Defense for Acquisition, op. cit., footnote 24, p. 18; see also: U.S. Depart.mentof Commerce, National Bureau of Standards, op. cit., footnote 77.

[84] Source: OTA Software Workshop. There are examples of civilian applications being designed and/or developed in Ada, but most commercial firms have adopted a "wait and see" attitude regarding the language. By comparison, Europear firms have elected to usc Ada in a variety of applications more frequently than their Us. counterparts have.

[85] Office of the Under Secretary of Defense for Acquisition, op. cit., footnote 24, p. 24.

[86] DoD-STD 2167A, Feb. 29, 1988, pp. iii/k.

[87] Ibid., pp. iii, iv, 3,4, 35, and 36.

efforts can begin.[88] The entire process is designed to guarantee that the government acquires the software that best meets its needs, ensuring that government funds are not misused or used for commercial benefit. The results of DoD's procurement strategy are contractual obligations that force commercial vendors to employ specialists fluent in the military regulations, government reviews, documentation, and accounting procedures required by the DoD. These regulations, audit requirements, and associated legal issues have forced many DoD contractors to establish autonomous divisions for conducting business with the government. Finally, vendors need a sufficient economic base to survive fluctuations in the DoD contracting and budget cycle.[89]

As a consequence, few civilian firms regularly contract with the DoD.[90] It has been argued that the limited base of contractors established to do business with the government inhibits DoD's ability to acquire quality software. While seeking the same quality software and the same assurance of a fair deal, the civilian software sector has no such regulatory mechanism. Performance, quality, and operational requirements of civilian software applications are weighed against cost. The commercial procurement process is designed to acquire the best software at the best price. Commercial-based contracts make no attempt to regulate or control the management practices of the developer, focusing instead on specification of the software functionality required. The numerous reviews and procedures required during the development of military software conflict with such a commercial-based practice.

The acquisition procedures and contracting practices used by the DoD not only limit the number of potential vendors, but discourage those contractors already qualified by the military. Civilian firms who contract with the DoD receive no guarantee of a continued relation with the DoD, accept poor profit margins, and often lose the rights in data to their software.[91] Government contractors therefore have little incentive to provide software that is innovative or of superior quality. The mechanisms used by the defense sector to select a software contractor in a sellers' market not only increase the chance that the DoD will get mediocre software, but frustrate many contractors from doing business with the military .92

Data Rights

The actual acquisition of software illustrates a further barrier to the transfer of software between the two sectors. Often, regardless of the software type, government contractors lose most, if not all, of their intellectual property rights to the software they develop.[93] The government's claim to unlimited data rights is based on the notion that these rights protect the government and will ensure public dissemination of publicly sponsored research efforts. In negotiating for unlimited rights in data for its software, the government gains the ability to maintain and modify its software systems in the future. Perhaps more importantly, this practice is intended to ensure the competitiveness of future

[88] Ibid., p. 9. The eight major activities are: System Requirements Analysis, Software Requirements Analysis, Preliminary Design, Detailed Design, Coding and Computer Software Unit Testing, Computer Software Component Integration and Testing, Computer Software Configuration Item Testing, and System Integration and Testing. These activities may overlap and may be applied iteratively. It should be noted that 2167A was developed to supercede DoD-STD 2167 and the waterfall lifecycle model it represented for software development. Yet the categorization of the development process in 2167A does not differ significantly from that in DoD-STD 2167; what it does allow is for the iterative application of the design and review processes in an attempt to accommodate a prototyping approach to systems development.

[89] Office of the Under Secretary of Defense for Acquisition, op. cit., footnote 24, p. 30. As an example, the time that elapses from requirements specification to letting a contracting to full deployment typically runs from 8- 14 years. By the time large defense systems software is deployed and operational, the computer hardware is 5 to 12 years behind that available on the market.

[90] The Defense Science Board Task Force on Military Software reported in 1987 that there were approximately 24 contractors regularly involved with mission-criticaJ software development for the DoD. More recently, International Resource Development, Inc., identified over 50 firms able to contract for large DoD software projects. In either case, these firms represent a minority of software firms in the United States. See: Office of the Under secretary of Defense for Acquisition, op. cit., footnote 24, pp. 29-32; and *Ada Data, International* Resource Development, Inc., Fall 1988.

9! Office of the Under Secretary of Defense for Acquisition, op. cit., footnote 24, pp. 29.

[92] OTA Software Workshop; also Office of the Under Secretary of Defense for Acquisition, op. cit., footnote 24, pp. 29-32.

[93] The government's *claim to* software *rights* in data are usually unlimited or restricted The former allows the government to "use, duplicate, disclose . . . software in whole or part, in any manner and for any purpose whatsoever, and to have or permit others to do so." For software developed wholly with private funds, the contractor can negotiate restricted data rights that give the government the right to modify software and make backup copies, but allow the developer to incorporate a typical licensing agreement with the government to protect lefforts. See: Michael Greenberger, "Rights-In-Data Policies Affecting Department of Defense Acquisition *of* Computer Software and Related Products," presentation for the Computer Law Association, Washington, DC, April 1988, pp. 4-6.

software maintenance and follow-on contracts. But according to many experts, in its efforts to foster fair competition, the government appears unable to compete effectively for its software. Further, many government employees and government contractors view the practice as onerous, burdensome, unnecessary, expensive and unfair.

Despite these acknowledgments and the flexibility allowed government contracting officers to negotiate less than exclusive rights to data in software acquisitions, the government usually insists on full transfer of data rights. In the commercial world, no company would demand exclusive rights to proprietary information, and many are dismayed by the government's expectation of these rights.[94] This practice clearly inhibits DoD's access to software developed by many civilian firms. The Institute for Defense Analyses Rights In Data Technical Working Group reported that because of the government's unlimited data rights demands,

> ... the government is failing to obtain the most innovative and creative computer software technology from its software suppliers. Thus the government has been unable to take full advantage of the significant American lead in software technology for the upgrading of its mission critical computer resources.[95]

The commercial sector typically protects proprietary information through laws relating to trade secrets. Contractual or licensing agreements govern the disclosure or dissemination of the intellectual content, or trade secrets, of software. Such licenses provide developers with continued revenue as they control the marketing of their product. In contrast, DoD's exercise of exclusive data rights does not guarantee the developer continued income or a further relationship with the government.. While the most recent DoD directives and regulations cite the ability and desire of the government to "negotiate" the rights to intellectual property, several factors have limited the practice or success of such negotiations. First, many DoD contracting officers and program managers intimate with the software contract do not have the guidance, knowledge, or experience necessary to request anything short of exclusive data rights. Second, the government generally receives exclusive rights to software that has been developed, either in part or wholly, with funds from the government. Third, developers who negotiate for restricted rights must meet government regulations and contractual obligations in order to Fully realize their rights.[96]

Ada

Some civilian software firms cite Ada as a barrier to working for DoD. The directive stating that Ada shall be the "single, common, computer programming language"[97] used in command and control, intelligence, and weapons systems may dramatically alleviate the military's software crisis. But because of Ada's relative immaturity, the number of commercial-oriented fires proficient in its use is limited. The mandate to use Ada appears to further decrease the already limited number of firms willing **and** able to contract with DoD.

Some experts cite Ada as an example of the government's tendency to standardize too many things too early. While the mandate to use Ada for mission-critical applications was arguably premature in 1983, developments associated with Ada weaken that argument.[98] But many commercial vendors, with the exception of those in the avionics industry, still have a wait-and-see attitude about Ada. While this is the prevalent strategy regarding Ada, there are successful examples of commercial Ada ventures, for example, in the development of

---

[94] Ibid.; also M. Greenberger, T. Shuba, J. Edmond, and R. Strassfeld, "Seeking the Balance Between Government and Industry Interests in Software Acquisitions" (SEI-87-MR-9), Software Engineering Institute, May 1987.

[95] Institute for Defense Analyses Rights In Data Technical Working Group, Draft Final Report, Nov. 22.1983. sec.1-1.

[96] OTA Software Workshop; also Michael Greenberger, op. cit., footnote 93; and M. Greenberger et al., op. cit., footnote 94.

[97] DoD Directive 3405.1.

[98] The original Ada mandate was issued as a memorandum from then Under Secretary of Defense, Richard DeLauer. But at the time of DeLauer's memorandum, there was a scarcity of validated, and more recently, performance-quality Ada compilers and development environments available in the industry, This situation has recently changed—the *Ada Information Clearinghouse reports that the* number of Ada development projects increased from 35 in August 1986 to 315 in September 1988. During the same period, the number of validated Ada compilers rose from 74 to 153, with an additional 65 compilers derived from these base compilers for similar hardware architectures. The market for Ada compilers, tools, environments, and training is growing at a compound annual growth rate of 35 to 40 percent and should exceed $750 million by 1990; see Jeremy Tennenbaum, op. cit., footnote 2.

compilers, tools, and in banking and communications applications.[99]

The merits of a single, standardized language will always be debated. In the case of Ada, the benefits include its embodiment of engineering techniques essential to the development of maintainable software; its support for modular (and reusable) components necessary in the development of large-scale, integrated systems; and increased portability among computer architectures. Additionally, because Ada was standardized early and trademarked, there are none of the incompatible dialects that have long been a problem in the software industry.[100] These characteristics may bridge some of the technological differences between the civilian and military sectors.

Whether Ada becomes an area of convergence, rather than a barrier, remains to be seen. Because the DoD remains the single largest consumer of software, and remains committed to the use of Ada, the language is potentially a major factor in future software technologies. Its potential, though, contrasts with the current situation where many military mission-critical applications are required to be implemented in Ada, while similar civilian applications will continue to be developed in the language thought best for that application.

Military Hardware Requirements

Requirements for hardened computers often result in the DoD buying specialized computers for some embedded and mission-critical systems. Given the close relation between hardware and software in these systems, this situation limits the potential number of vendors who can develop software for these applications. It is particularly evident in the Navy, which typically contracts for special-purpose, non-commercial, hardware.[101] These specialized hardware requirements exacerbate the incompatibility that exists among many software systems with similar applications. Barriers of incompatible interfaces, languages, operating systems, and protocols created between militarized hardware and commercial hardware architectures make it less likely that any transfer can or will occur between the defense and civilian software sectors.

[99] Source: OTA Software *workshop:* also: John Burgess, " 'Universal' Computer Language Finally Takes Hold at Pentagon," ***Washington Post,*** July ***17, 1988, pp.*** H1, H5.

[100] Ada was originally trademarked to prevent the creation of "supersets" (extensions) and "subsets" of the language—all implementations of the language must meet MIL-STD 1815A fully to be considered Ada. This, combined witb the requirement that all Ada compilers pass a validation (conformance) process, helps ensure that Ada is portable across computer architectures.

[101] Based on the Navy's Next Generation Computer Resources Program briefing to OTA.