Routledge
Taylor & Francis Group

The Information Society
An International Journal

ISSN: 0197-2243 (Print) 1087-6537 (Online) Journal homepage: http://www.tandfonline.com/loi/utis20

American Hegemony in Packaged Software Trade and the ''Culture of Software''

ERRAN CARMEL

To cite this article: ERRAN CARMEL (1997) American Hegemony in Packaged Software Trade and the ''Culture of Software'', The Information Society, 13:1, 125-142, DOI: 10.1080/019722497129322

To link to this article: http://dx.doi.org/10.1080/019722497129322

American Hegemony in Packaged Software Trade and the "Culture of Software"

ERRAN CARMEL

Kogod College of Business Administration
American University
Washington, D.C., USA

United States-based companies continue to dominate the global market in packaged software. Although many have predicted this U.S. dominance will wane, particularly as a result of competitive threats from Japan, and more recently from low-wage, developing nations, erosion has yet to be significant. The United States benefits from nine factors that sustain its advantage in this industry: skilled labor, favorable capital conditions, sophisticated customers, close association with hardware vendors, a competitive marketplace, geographic concentrations, first-mover advantage, a strong intellectual property regime, and English as the software lingua franca. Industry-specific strengths and weaknesses vis-à-vis Europe, Japan, and other nations are also discussed. In addition to these nine better known U.S. competitive advantages, two culturally linked assertions are presented that have received scant attention vis-à-vis competitive analysis. First, the industrial evolution of software development is at an immature stage—still a cottage industry practiced by craftsmen in a cultural milieu of artisans—and thus does not track other global high-technology trends. Second, packaged software is part of the copyright industry (e.g., film and music) in which United States-based firms have a sustained advantage. While manufacturing capabilities are significant for technology industries, culturally related factors, such as creativity, are more important for copyright industries. The U.S. "culture of software," which helps explain U.S. hegemony, is introduced and discussed. The three elements of this culture are the culture of individuals as manifested by the individualistic computer hacker; the entrepreneurial culture and its risk-taking ethos; and the software development culture with its embrace of ad hoc, innovation-driven development as opposed to routinized, production-driven development.

Keywords commercial software, competitiveness, Europe, hacker, intellectual property, Japan, software development, United States

America's overall market leadership [in packaged software] is not 'grunt work' that can be done at low cost, but is mostly design work—Umang Gupta, Indian-born founder and CEO of California-based Gupta Technologies, a large packaged software vendor (Price Waterhouse, 1993, p. 14).

Received 20 August 1995; accepted 18 December 1995.

The author thanks the following for valuable suggestions in preparation of this article: Jason Dedrick, Frank DuBois, Kathy Getz, Sy Goodman, David Jacobs, Renee Marlin-Bennett, Jack McPhee, Paul Quintas, Steve Sawyer, Stephen Siwek, Kathy Zettl-Schaffer, the anonymous reviewers, and the many software developers and managers on three continents who have shared their experiences. Early drafts of this article were circulated with the titles "Can the US stay ahead in packaged software?" and "Software as culture—The ignored explanation for US competitive advantage in packaged software."

Address correspondence to Erran Carmel, Kogod College of Business Administration, American University, Washington, DC 20016-8044, USA. E-mail: ecarmel@american.edu

The Information Society, 13:125–142, 1997

0197-2243/97 $12.00 + .00

United States firms lost their global dominance in electronics and automobiles as manufacturing migrated to nations that were able to manufacture these products cheaper and, later, better. In packaged software, however, the United States has sustained a dominant position against formidable global contenders. What brought about this American hegemony and is it likely to continue? In order to address this question, this article comprehensively discusses U.S. competitive advantages in packaged software including the "culture of software," which has not received the serious attention that the subject merits.

Because software is an emerging, amorphous industry, it is important to define the three principal domains in which software is developed: internal development, software services, and packaged software.[1] Most software is still developed internally—within organizations on a custom basis—such as information systems for business, government, or military. Such software is not traded. In contrast, the other two software domains are tradable and competitive, domestically and across national frontiers. Software services encompass systems integration and consulting (including project management, requirements analysis, design, contract programming, education, training, ongoing system support, and maintenance). Software service providers can range from an individual custom programming consultant to large globally competing software services firms such as U.S.-based Electronic Data Systems and French-based Cap Gemini Sogeti.

The packaged software[2] industry traces its roots to the 1968 U.S. Justice Department's decision to force IBM to unbundle hardware and software sales. Over the years organizations have learned to reduce their dependence on expensive, custom software and purchase packaged software. Diffusion of the microcomputer in the 1980s led to a boom in the industry. *Packaged software*[3] is defined to include all software sold as tradable products (purchased from a vendor, distributor, or store) for all computer platforms including mainframes, minis, and microcomputers. Packaged software is typically categorized into "systems" versus "applications" (more recently, other important categories include "tools" and "content").

Table 1 and Figure 1 illustrate the dominant competitive position of the United States in packaged software. The global software package market is estimated to be $77 billion (U.S. Department of Commerce, 1995), with U.S. market share estimated

Table 1
Position of major blocs in the packaged software industry

	United States	Europe	Japan	Sources
Number of firms in Software 100—1994	85	11	0	Frye and Melewski (1995)
Number of firms in *Datamation*'s Software Global-15 (includes hardware firms' packaged software sales)	8	4	3	Datamation 100 (1993)
Packaged software revenues as percent of total software industry revenues (data for 1990)	40	38	9	Schware (1992)
Trade surplus/deficit in packaged software	Positive, but NA	−$18 billion	NA	*Economist* (1994)

Note. NA, not available.

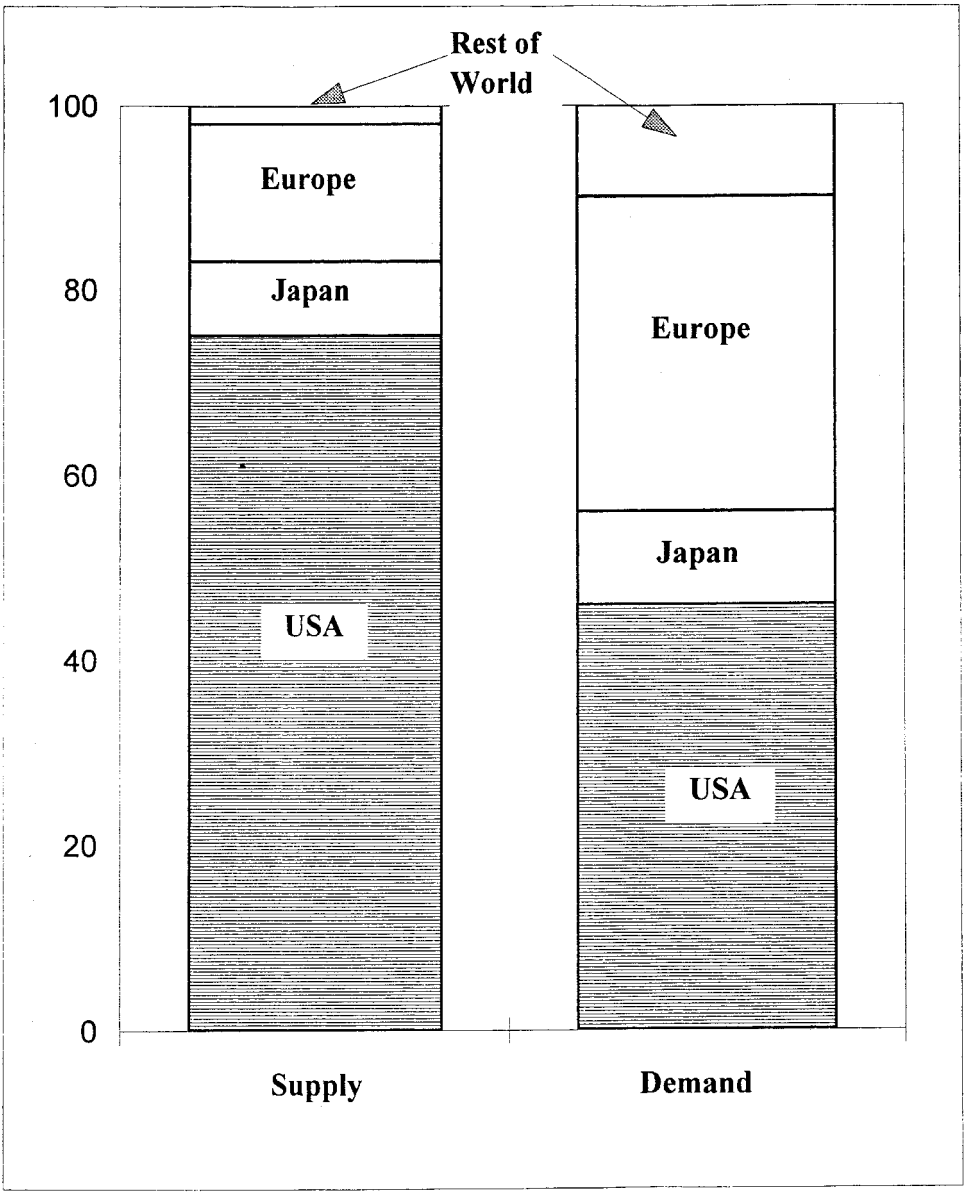

Figure 1. Supply and demand of packaged software (U.S. Department of Commerce, 1995; *Economist,* 1994).

to be 75%. With such a strong international competitive position, the United States and the U.S. packaged software industry have much to lose. Indeed, U.S. analysts, pointing to historical precedence in other industries, have oscillated between predicting the triumph of foreign software (e.g., Cusumano, 1991; Jones, 1994a; Yourdon, 1992; Zuconni, 1990) and reveling in the failure of foreign, particularly Japanese, competition to catch up to the United States (e.g., Clark, 1992; Hamilton, 1993; Schrage, 1993).

The most commonly cited competitive threats to the U.S. software industry are those from Japan and from low-wage, developing countries. The competitive threat from Japan is perceived as its ability to perfect technologies and is best illustrated in its software "factories" (Cusumano, 1991). The competitive threat from low-wage countries, labeled "off-shore programming" (cf. Ravichandran & Ahmed, 1993), at this writing, is of greater concern in the United States. Much of that concern has focused on Indian firms, which have been successful in the 1990s in contracting with dozens of American firms to perform software development. Not only have the Indian operations offered cost advantages, but some are now highly successful in quality. In 1994, an Indian organization was one of only two organizations in the world to attain the world-class SEI level-five ranking (Gibbs, 1994).

Over the years a number of reasons have been posited as to why the United States is expected to lose its dominance in software:

- Labor costs. Relatively high U.S. labor costs are already driving software to off-shore programming. Unlike manufacturing, software requires little capital infrastructure and is characterized by low startup and operations costs.
- Productivity. Some argue that U.S. programmer productivity is no higher and perhaps lower than that in other nations.
- Technology transfer. The rapid diffusion of software knowledge due to permeability of national borders hurts the United States, with its open information sources.
- Government support. It is widely recognized that software is a strategic sector because it fosters innovation in the economy and because of military significance. The U.S. government has been less supportive than foreign governments, particularly in Japan and Europe, in supporting the software industry[4] (Brown et al., 1995; Siwek & Furchtgott-Roth, 1993; Zuconni, 1990; also see Table 2, item 8).

However, these arguments for near-term competitive decline have proven to be largely incorrect, insufficient, or premature as the remainder of this paper contends. The next section presents nine competitive advantages that partially explain U.S. hegemony in the software industry in general and packaged software specifically. This is followed by a discussion of competitive factors vis-à-vis Japan, vis-à-vis Europe, and vis-à-vis other countries. Our fourth section introduces two additional topics that highlight the significance of the culture of packaged software in the fifth section.

U.S. Packaged Software Industry's Competitive Advantages

This section attempts to assemble U.S. competitive advantages in the packaged software industry that are commonly mentioned in related analyses. Some of the analytical framework is due to Porter's model of competitive advantage of nations (1990), and several major points are due to Siwek and Furchtgott-Roth (1993). Table 2 displays some key figures that shed light on various points raised in the discussion that follows.

The nine competitive advantages are:

First-Mover Advantage. The software industry was born in the United States and benefited from a combination of supportive demand conditions (defense) and supply conditions (hardware manufacturers needed software to sell computers). As a result, industry standards emanated from the United States and the lingua franca of computing became English. Later, the standards of the microcomputer revolution (IBM PC, Apple, Mi-

Table 2
Comparative figures that help explain competitive outcomes

Item	Parameter	United States	Europe	Japan	Sources
1	Traded equity portfolios with investments in startups	14%	3%	NA	*Economist* (1994)
2	Venture capital investment in computer-related entrepreneurial firms, 1993	$2400m	$300m	NA	*Economist* (1994)
3	Percent of 20–24 year olds in tertiary education, 1990	72%	33%[a]	31%	Brown et al. (1995)
4	Percent of primary and secondary schools with computers	90%	40%	30%	Brown et al. (1995)
5	Number of software users	23m	20m[a]	9m	Jones (1994b)
6	PCs per 100 workers	63	30[b]	20	Koertz (1994)
7	Software piracy rates, 1993	35%	60%[a]	80%	Business Software Alliance (1994)
8	Government funding for major software-related R&D programs since 1980	$730m	$1900m	$2600m	Brown et al. (1995)

[a]Extrapolated/averaged by this author.
[b]Germany only.

crosoft) and, more recently, the Internet were U.S. developed. First-mover advantage led to a large, installed base of U.S. packages around the world. First mover in software is critical because there are few products that so thoroughly shackle a customer to a vendor because of the investment in training, setup, and customization.

Factor Conditions: Labor. The United States is well endowed with flexible, skilled, and educated labor trained in computers. The number of math and computer doctoral students in the United States has been rising steadily. Although many are foreign students, many of these students stay in the United States upon completion of their degrees. Unlike manufacturing firms, most U.S. packaged software firms develop domestically and often import highly skilled programmers from around the world. Although U.S. wages are high, compared to those of programmers in developing countries, Schware (1989) argues that these wages reflect higher productivity.

Factor Conditions: Capital. The U.S. capital markets are supportive of software innovation with specially targeted software venture capital funds and easy access to securities markets (Greenbaum & Johnston, 1993). Hundreds of millions of dollars in U.S. venture capital are specifically targeted at software (Table 2, items 1 and 2).

Demand Conditions. Sophisticated and demanding customers spur innovation (von Hippel, 1986). American customers use the largest number of function points (units of software capability) per software user (Jones, 1994b). Broad sectors of the U.S. population have been exposed to computing as indicated by Table 2, items 3–6. The U.S. business

and household customers demand the latest software and communicate these needs to vendors. The geographical proximity of vendors and customers has accelerated product cycles, further impeding foreign entrants not closely tied into these customer–vendor innovation loops.

Related and Supporting Industries. The United States is the world leader in most segments of computing including hardware, software, and many areas of telecommunications. These segments interact and influence each other. Even within the packaged software industry itself, there are value chains of firms that are mutually supporting in which benefits flow to firms horizontally and vertically. For example, the major operating system companies are U.S.-based. These companies interact with a large population of programming tools companies, which, in turn, interact with applications developers. Still newer segments are becoming similarly intertwined with software such as U.S.-based entertainment firms.

Competitive Environment. The U.S. packaged software companies operate in a fiercely competitive market for "talent" (labor), market share, and exposure. At the same time companies continue to develop complementary and interlocking software that works seamlessly with products of their fiercest rivals. Rivalry drives up software employee compensation with possible windfall profits (e.g., three thousand "paper" millionaires work at Microsoft), which is fueled by capital pools that invest in innovative startups. The packaged software industry has been completely "exposed" to competition and received no "sheltering" from government (a sheltered industry enjoys national protection or guaranteed large government purchases, as in defense industries). Derian (1990) argues that "exposed" industries are more successful in the long run. Roberts (1991) researched American high-tech firms and found that those that "productized" software earlier and more quickly had higher rates of success.

Geographic Concentration. Technological innovation tends to occur in geographically concentrated areas, where formal and informal networks of companies and individuals interact to create a special cultural milieu. Such is the case in California's Silicon Valley, Boston's Route 128, and several other smaller, high-tech corridors, which many other nations have tried to reproduce. The Silicon Valley model of culture is one in which its principal actors—executives, engineers, programmers, and technicians—share a common set of values and lifestyles, forming a milieu of innovation (Castells & Hall, 1994).

Intellectual Property Regime. Strong intellectual property protection and enforcement in the United States results in one the world's lowest software piracy rates (see Table 2, item 7). Other nations' intellectual property regimes are weak, particularly in developing nations, resulting in piracy rates as high as 99% (Business Software Alliance, 1994).

English. English is the lingua franca of computing, the second most widely spoken language in the world, and one of the most popular as a second language. Furthermore, measured in income dollars, the English-language market is the largest in the world. Language, in turn, has hampered the United States' principal technology rival—Japan. The Japanese language, with its thousands of ideographs, requires complicated keyboards and complex software.

Comparative Examination: Japan, Europe, and Other Nations

The competitive analysis is now extended to specific comparisons of the U.S. packaged software industry to those in Japan, Europe (treated as a bloc), and in other nations.

United States Versus Japan

With a strong domestic computer industry, perhaps the strongest potential challenge to U.S. software comes from Japan.[5] Major Japanese computer hardware companies are among the world's top software producers (Table 1), and domestic vendors dominate markets in custom software services. Japan has a proven track record in product development approaches (i.e., concurrent engineering, close relationships to customers and suppliers, incremental improvement paradigms, adapting research into production). Contrary to those who say that the Japanese have not been creative, a great deal of creativity is reflected by their success in computer games such as Nintendo (Neff, 1995). The current convergence of hardware, software, and consumer electronics into multimedia may play favorably to Japanese strengths. Finally, the Japanese culture is more conducive to software development in teams than American culture, which is too individualistic to foster proper teams.

Japan is also notable for government-sponsored national computer and software initiatives beginning with the Fifth Generation Project (artificial intelligence) in 1982, TRON (operating systems) in 1984, Sigma (reusable code) in 1985, FACET (software engineering), and others.

In spite of these factors, some analysts position the Japanese software industry 3–5 yr behind that of the US. Many ex post facto explanations surfaced to explain this paradoxical outcome.

Cottrell (1994) attributes much of Japan's lag in microcomputer software to a fragmentation of software standards that confused both the development and the user marketplace. The Japanese computer industry gambled on imposing a Japanese operating system standard that ultimately failed, delaying Japan's entry to global software markets for U.S.-imposed operating system standards. Microcomputer diffusion was also hurt by high prices resulting from Japan's idiosyncratic distribution and market structures. Furthermore, the Japanese computer hardware industry, unlike its American hardware counterparts, has been slow to decouple software from hardware. This has stifled the development of independent software providers in Japan.

Clearly, the Japanese language has been an obstacle to mass computerization. The thousands of characters in kanji cannot fit on a keyboard and require more elaborate schemes for typing (Mori & Kawada, 1990).

While sophisticated U.S. business users have spurred the U.S. software industry, the Japanese business sector has not had a similar impact. Japanese industry prefers custom software to packaged software, and crowded Japanese offices have little desktop space—delaying wide dispersion of PCs during the 1980s. Perhaps more importantly, a PC on a manager's desk has not been a mark of prestige. Less than 1 in 10 Japanese executives feels that a computer is essential to his job (Fox, 1995).

Some aspects of the Japanese software development culture may also be a competitive hindrance. First, programming is not a respected career in Japan, leading to a shortage of skilled programmers. Second, some point to a lag in embracing the object-oriented approach (Schwartz, 1992). Finally, some argue that software requires special creativity whereas Japanese firms have relied heavily on copying tools and techniques, placing too much emphasis on an engineering orientation.

United States Versus Europe

The advanced industrialized countries of Europe have a well-developed independent software industry that has been supported by national and European Union (EU) software research programs (the EU has invested more than one billion dollars in the Esprit program begun in 1984; see Table 2, item 8). However, like their Japanese counterparts, European firms have focused on custom development more than packaged software. Four European nations are major global exporters of packaged software: the United Kingdom, France, the Netherlands, and Germany. However, in all these countries U.S. software companies continue to dominate domestic markets (Hudson, 1994).

Relative to the United States, Europe has several structural disadvantages that affect packaged software (Greenbaum & Johnston, 1993; *Economist,* 1994). Perhaps most important is the capital market structure. European venture capital is smaller and less generous (Table 2, items 1 and 2) and there is no NASDAQ-like secondary market to provide capital to small businesses. During my own visits to European packaged software firms, I frequently heard grumblings about poor venture capital sources (Carmel, 1995). With little venture capital, many small companies turn to banks that do not have the managerial expertise to assist young companies. In spite of the Maastricht treaty, national borders and cultures act as trade barriers and the EU market is still fragmented. Many young European companies stay small because their domestic markets are small. Furthermore, there is no critical mass of programming talent anywhere in Europe that resembles that of Silicon Valley (the M-4 corridor west of London comes closest). Entrepreneurial activity is less respected, driving some enterprising Europeans to the United States.[6] Finally, fewer European entrepreneurs have management training because there are fewer European business schools.

Relative to Japan, Europe has been more successful in producing small entrepreneurial high-technology firms. Clearly, the Europeans have one advantage over the Japanese—that of language. Not only do more Europeans have a greater command of software's lingua franca, English, but their closely related languages can support the ASCII character set on keyboards.

United States Versus Other Developing Countries

Developing countries present the classic comparative advantage first proposed by Ricardo: low labor costs. The low wages in these countries are often combined with good computer training and high productivity: Jones (1993) estimates that the lowest cost per function point is in growing, educated markets such as Mexico and India. Additional nations on this list include Philippines, Bulgaria, and Russia (cf. Press, 1993; Nidumolu & Goodman, 1993). In India alone there are an estimated 1.4 million software professionals, and in 1993 the country exported nearly half a billion dollars in software (Taylor & Guptara, 1995). But in spite of the growth of these software centers, neither India nor other offshore programming nations have offered a competitive threat to American (or European) packaged software companies' dominance. For example, India's recent packaged software exports totaled roughly $50 million (Taylor & Guptara, 1995), representing a tiny fraction of a percent of global packaged software sales.

United States Versus Other Developed Countries

Some small developed nations, where labor costs are at roughly U.S. levels, have been successful in developing strong software industries. Dedrick et al. (1995) list Singapore,

Hong Kong, Ireland, Israel, New Zealand, and the Scandinavian nations in this category. Each of these small nations has traveled the path to national software success for very different reasons. All have established a strong foundation of computer professionals and a sophisticated customer market. Although some of these nations have been successful in exporting packaged software, none poses more than a nuisance threat to U.S. global dominance in packaged software.

Peculiarities of the Packaged Software Industry

This section explores two pecularities of packaged software that allow us to delve deeper into the question of sustained U.S. hegemony in packaged software. While the nine competitive advantages (discussed in the previous section) are either structural (in the economic sense) or have been widely discussed elsewhere, these two topics have not yet been integrated into competitive analysis. The first topic, the industrial evolution of software development, leads to the assertion that since software is an immature industry, it will not track other high-technology global competitive trends in the near future. The second topic, packaged software as part of the copyright industry, leads to the assertion that packaged software inherits some U.S. competitive advantages in copyright products. These two topics link to the discussion of the software culture in the following section.

The Industrial Evolution of Software Development

Evolutionary stages of industrial product maturity follow a predictable pattern that usually spans many decades. Beginning with the emergence of a craft, an industry then standardizes parts and, in parallel, standardizes the process of production (Kogut, 1991). As these stages unfold they lead to shifts in national advantages. Following Scherer (1992), the sellers of differentiated products in early life cycle stages gain some degree of monopoly power through economies of scale, pricing and distribution strategies, or skewing the terms of trade in favor of national producers. Indeed, Scherer's normative model is manifested in the packaged software industry to date. Over time, however, the knowledge to manufacture the parts and to set up the process is diffused through technology transfer. Today's permeability of borders has quickened the shift in gaining and relinquishing country advantages (Kogut, 1991).

Yet software is still in an immature stage of its industrial revolution. Significant sectors of the software industry are still a cottage industry operated by artisans. Software development is still practiced as a *craft*—by intuition and experience. Artisans build everything from the ground up—not only their software code, but the processes, methodologies, and experience sets. Software development has witnessed many failures, both large and small, that have led to the never-ending "software crisis" and the classic problems of late, overbudget, and buggy software. Mary Shaw (in Gibbs, 1994) positions today's software development in the maturity stage analogous to that of chemical engineering in the early 1900s. Today, chemical engineering has reached the stage of professional engineering in which progress relies on scientific principles. In contrast, software has just begun to move into an intermediate maturity stage of early scientific refinement and commercialization.

The term *software engineering* was introduced in 1968 and is defined as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" (Gibbs, 1994). Today, efforts to base software development on scientific and engineering principles represent a broad movement including es-

tablished benchmarks (e.g., the U.S.-based Software Engineering Institute's model), automation (e.g., Computer Assisted Software Engineering), and government initiatives (e.g., the United Kingdom's Alvey project).

In some respects software is following the historical trajectory of industrial products. Maturity of parts and maturity of process are taking place through two, somewhat different, paths. Software parts are referred to today as "objects" or "components." The notion of software objects facilitates a market in which programmers buy and sell these objects and use them to quickly build software systems. Separately, the discipline of software engineering, and its related schools, is pushing software up the process maturity curve. This could turn programming into a true engineering discipline in which the stages and tasks are well understood, documented, and repeatable rather than ad hoc.

If the historical trajectory is deterministic, then why have the Japanese not yet succeeded at software? After all, Japan has repeatedly succeeded in assimilating technological processes and products and perfecting their manufacture. Japanese firms absorb manufacturing process models, refine and improve them, and then are able to create products of high quality with rapid time-to-market. Outside of the software area, Mansfield (1988) found that while American firms devote about two-thirds of their R&D expenditures to improved product technology (new products and product changes) and the remaining one-third to process technologies (new processes and process changes), in Japanese firms the proportions were reversed—two-thirds of spending is process R&D. We see this phenomenon reflected in Japanese software development. The Japanese have perfected the software development process into so-called software factories, allowing rapid, high-quality production. Japanese companies excel at software techniques and technologies (e.g., project management, code reuse, and tool usage), demonstrate higher productivity (Cusumano & Kemerer, 1990), and generally have higher quality production (Brandt, 1991).

In summary, Japan has not succeeded in packaged software, in part, because the state of the art in software development is still too immature. It follows that, once the software process matures, Japan will be able to enhance software development and compete successfully because of its superiority in process technologies. However, such a scenario may be less likely, in part, because packaged software is not simply a technology product, as I argue next.

Packaged Software as Part of the Copyright Industry

The copyright industry (including newspapers, books, radio, television, video, film, theater, advertising, as well as software) makes up 3.74% of the U.S. gross domestic product (GDP) and represents the second largest U.S. export category after autos and auto parts (Siwek & Furchtgott-Roth, 1995). There is no question that the United States has had a sustained competitive advantage in copyright products. Partially because of this, U.S. postwar hegemony in informational, cultural, and iconic elements has been labeled "cultural imperialism" (even packaged software has recently been so labeled; Quintas, 1994). To date, there is little understanding of how to apply the concept of competitive advantage to the copyright industry. For example, Porter's seminal work of 1990 has but short anecdotes about the motion picture and software industries and makes no attempt to group these industries in a separate category. It is likely that Porter did not anticipate the sudden emergence of intellectual property trade as a global issue.[7]

Packaged software and other copyrightable products share two related cultural characteristics pertinent to this analysis. First, copyright products are not indifferent to language. Specifically, English, the lingua franca of computing and business, is also the

dominant language of other copyright industries. Second, to some degree, the culture of the home country is embedded in the product itself. America is often the barometer for global tastes such that success in the United States translates to global diffusion (e.g., there are many beautiful, talented performers in every nation, but none can hope to attain the success of American pop singers such as Madonna or Michael Jackson). Multimedia, the blending of sectors within the copyright industry, builds on existing U.S. competitive strengths (e.g., the collaboration between Hollywood and Silicon Valley is labeled "Siliwood").

Additionally, packaged software and other copyrightable products share production characteristics pertinent to this analysis. Although all these products often share a need for advanced technology to manufacture the first copy of the product (e.g., fast microprocessors, video equipment, music synthesizers), firms in the industry do not compete on the basis of manufacturing capabilities. Rather, it is the innovative, creative component of development—and not advantages in cost and engineering innovation—that often determines success and competitiveness. By extension, copyright products may not follow historical trends of manufactured products in which process improvements shift production to low-wage nations.

Both topics in this section are linked by cultural elements. The young, immature packaged software industry, described in the first part of this section, is one in which the dominant cultural milieu consists of artisans and skilled craftsmen rather than scientists and engineers. At the same time, global trade in copyrightable products (such as packaged software) takes place in a cultural milieu in which the United States and its native language have a dominant impact on global cultural forces and tastes.

The American Culture of Software

While cultural analysis is not new, it has rarely been applied to industry-specific international competitive analysis. An exception is the seminal research of Hofstede on national cultural norms and their application to management and organizational questions (1980, 1993), recently applied to the case of airlines and airline safety (Phillips, 1994).

Culture refers to a set of patterns, behaviors, and symbols representing the ideas and values of a group. It is a construct that is used to describe nations, ethnic groups, and tribes. More recently, researchers have developed a deeper understanding of new cultural groupings such as organizational and corporate cultures. A further extension of cultural groupings is the notion of a subculture—binding individuals who share a common interest or profession.

Thus, the American culture of packaged software is an intersection of characteristics unique to the (national) American culture and of the subculture of computer programmers and software professionals who share values, interests, and life-styles relating to computers and software.

Yet, this culture is not an ethnocentric culture. Many of those who succeeded in the American software culture, such as Umang Gupta, founder and CEO of California-based Gupta Technologies, are themselves not American born. The continued infusion of foreign-born entrepreneurs into this industry is illustrated by a recent listing of "hot" young U.S. software companies (Upside, 1995). The list includes numerous Asian names among the companies' founders and owners. These talented, educated immigrants self-select into the American culture of software and assimilate many of its values.

In packaged software development, three elements form the American culture of software: the culture of individuals, the entrepreneurial culture, and the culture of soft-

ware development. These three elements are presented next, followed by a brief discussion of Microsoft as an embodiment of the culture of software.

The Culture of Individuals

Certain personality types are drawn to packaged software companies and define their dominant culture. Castells and Hall (1994) label the Silicon Valley culture as one of extreme individualism. Americans believe in the myth of individualism, in which the hero is a maverick, or rebel—someone who stands out and does things his or her way. Hofstede's data (1980) confirm this, ranking the United States highest in individualism of 40 countries in his study.

The myth of individualism bred a computer-related spin-off: the "hacker." Although often criticized, hackers are seen as heroes in American contemporary popular culture.[8] A definition of *hacker* was compiled from many formal sources (most notably from Kyle, 1994) and informal sources, as a person with some, or all, of the following qualities:

> Thinks of self as a software wizard or a software artist; depends on wizardry to mitigate the fundamental limitations of software; able to write code quickly under pressure; takes an experimental approach to software development; without formal training in software development; lacking in discipline; whose hobbies and work have to do with computing; writes software for fun, challenge, and learning.

Famous hackers have been anthologized in books (described as "superstar programmers" in Lammers, 1986, and "wizards" in Levering et al., 1984). The most famous hacker, Bill Gates, founder of Microsoft, has been the subject of countless essays and books. Hackers are considered fashionable, illustrated by the success of the American computer pop-culture magazine "Wired." The American individualist hacker stands in marked contrast to descriptions of Japanese programmers as overly conformist (Watanabe, 1990). Cringely (1992) calls the founders of the microcomputer industry *adolescents* that split off and started their own culture. He recounts a product launch celebration at Adobe Systems, Inc., one of the largest U.S. packaged software firms, in which all 280 employees armed themselves with wastebaskets for a company-wide waterfight.

The hacker's computer obsession represents both a vocation and a life-style that defines his or her social life, peers, and communication style. A well-known legend that emerged in the culture is that of late-night work hours. This, in turn, has led to mythologized culinary habits of eating junk food from vending machines. Dvorak (1993) states that "the best code is still done by small teams of young American coders eating pizza and drinking Coca-Cola." Cringely (1992) sees these (and other) adolescent habits as important to understanding the success of the software culture in a rapidly changing marketplace, arguing that by comparison the Japanese are "too grown-up, too business-like, too slow."

Interestingly, many individuals in packaged firms do not have computer-related degrees. My own data gathered in U.S.-based firms indicates some hackers, attracted to computers at a young age, have little tertiary education and occasionally have not even completed their secondary education. These are generally individuals who are somewhat rebellious toward the mainstream social system including its educational institutions.

Yet the ethos of the software hacker is one of respect for achievement, as represented by more efficient code, a sophisticated program, or, when ethical boundaries are violated,

a new computer virus. Building and creating are at the center of the hacker's life. Building and experimenting with software is the primary focus of both professional and social activity, blurring the boundaries between the two.

In summary, a number of social factors have fostered a sustained and successful cultural ethos: the myth of the software hero, the trappings of a hacker subculture, the behavioral norms (e.g., rebelliousness), all coupled with a respect for achievement.

The Entrepreneurial Culture

Although most of the individuals in the U.S. packaged software culture do not run their own companies, entrepreneurialism serves as one of the culture's icons. Respect for entrepreneurialism is inherited from the broader American culture that has mythologized the entrepreneurial hero (Reich, 1987). In contrast, the European or Japanese entrepreneur is not as well respected as his or her American counterpart.

An entrepreneurial culture stresses competitiveness, risk-taking, independence, and creativity. Competitiveness reflects a free-market, aggressive outlook that honors those reaping fortunes (the U.S. archetype is the fictional Horatio Alger who rose from poor beginnings into wealth). Risk-taking is deeply rooted in American culture: Hofstede's data (1980) rank the United States low on uncertainty avoidance, implying a high degree of risk taking. Independence, creativity, and innovation are all associated with entrepreneurship: Pacey (1983) finds that in large organizations, the innovators are driven out of large organizations to join, or establish, small firms.

In many ways packaged software companies represent the pinnacle of modern American entrepreneurship. Software is an industry that was largely built by startups begun in garages or small offices: In the period 1980–1984, 2026 U.S. software firms were founded (*Business Week,* 1994). The U.S. investment community has also shown considerable respect for the entrepreneurial software culture by continuing to encourage young software startups: Software companies are now going public at the relatively young stage of $23 million in annual revenues (*Software Magazine,* 1994).

The Software Development Culture

The software engineering paradigm is rejected by many in the American software culture, much to the frustration of some of its advocates within the professional community. In the meantime most U.S. packaged software development has not emerged from the craft stage into the professional/engineering stage.

The rejected engineering-oriented discipline is displaced by an emphasis on innovation, creativity, and a rapid prototyping approach to development (which is part of a larger trend to a "prototyping culture" in U.S. industry; Peters, 1995). Innovation and engineering are seen as contradictory: innovation, a key determinant of packaged software success, cannot flourish in top-down factory approaches that place an emphasis on documentation, controls, and data collection. Thus, it is the bureaucratic, restrictive nature of software development, rooted in software engineering approaches, that is least conducive to a successful software industry.

Innovative software developers should be encouraged to "break the rules." The president of U.S.-based Sun Microsystems, a large hardware and software vendor, points out that "[the development] process is best practiced slightly out of control" (Brandt, 1991). This tone is reflected in more scholarly sources, as well: "the profession must reconcile the different image of software engineering around the world. The creative white collar

image prevalent in the U.S. clashes with the grind-it-out, blue-collar image that prevails in Japan" (Chang & Burgess, 1991).

Since the American software culture is partially premised on notions of innovation and creativity, it is important to examine creativity in the national context. This is a difficult and controversial construct to operationalize. A common creativity determinant is thought to be diversity of backgrounds and ideas. Following this logic, the United States possesses a number of factors that contribute to creativity: breadth of education and the openness of software industry. First, it is argued that a broad education is most likely to foster creativity (e.g., Brown et al., 1995). The United States has one of the highest ratios of adults with some tertiary education that includes a high ratio of course diversity. By inference, the American university-trained software professional is better rounded. Second, the breadth of ideas in the industry may also be explained by the software field's openness to individuals who were not formally trained in computer fields: Many are from non-computer fields such as the physical sciences and other engineering disciplines, as well as the social sciences and liberal arts.

The creativity debate is crucial because the argument has been made that the Japanese culture of conformity is less conducive than the U.S. culture to innovation in software development. This argument is even taken seriously by the Japanese government (Brown et al., 1995). An illustration of this difference can be found in a study comparing U.S. and Japanese technical personnel (Sakakibara & Westney, 1992), which found that U.S. personnel are more likely than their Japanese counterparts to take additional educational courses to acquire new skills (rather than update existing skills) and are more likely to do so in order to improve their chances of being assigned to more interesting activities.

The Case of Microsoft

Microsoft is an interesting manifestation of the culture of software because it embodies many of the attributes of the culture (and has been chronicled extensively; Alpert, 1992; Cusumano & Selby, 1995; Kosnick, 1990; Lammers, 1986; Zachary, 1994). The case of Microsoft is important to this analysis in two respects. First, it is an American firm with success and hegemony in packaged software that is incomparable to any other firm. It commands approximately one-third of the global desktop packaged software market and has the largest deployment of software (roughly equivalent to diffusion) in the world (Jones, 1993). Second, due to its success and notoriety, its cultural and management models are consciously emulated by many U.S. and non-U.S. packaged software companies.

All three elements of the packaged software culture are present at Microsoft. First, many of Microsoft's employees, including its CEO, have hacker characteristics and share the individualist values of that culture. Microsoft's developers can express their free-spiritedness in casual dress, unusual office decorations, and flexible work schedules. Their lives revolve around long work weeks, and many aspects of their social life and hobbies revolve around computers and Microsoft. Microsoft programmers view software development as "a craft, like wood-working or masonry" (Alpert, 1992), and stress individual responsibility. Second, although the company has grown immensely and is no longer a small entrepreneurial firm, it still tries to retain that ethos through its team structures. Finally, the software development process is still one that stresses the craft of software: The developer is still the star, now aided by a personal tester to keep the developer from submitting compromising bugs (Cusumano & Selby, 1995).

Summary

This article has broadened the perspective regarding the sustainability of U.S. hegemony in packaged software. After presenting nine initial competitive advantages, the following two additional assertions were posited. First, software is relatively immature as a technological discipline. Thus, packaged software is still a cottage industry, practiced by artisans, in which the culturally driven forces of a craft dominate production capabilities. Second, packaged software trade needs to be seen within a constellation of the larger copyright industry, rather than simply as a technology industry. For copyrightable products, creativity-related factors dominate manufacturing considerations. The "culture of software" links these two topics. Specifically, the American culture of packaged software, derived from the national culture and the computing subculture, is composed of three elements: the culture of individuals as manifested by the computer hacker; the entrepreneurial culture and its risk-taking ethos; and the software development culture with its embrace of innovation-driven versus production-driven development.

Two major qualifications to the cultural thesis introduced in this article are (1) that the culture has negative, unproductive characteristics, and (2) that the culture is not uniquely American.

Numerous high-profile development failings suggest that fundamental problems exist in the U.S. packaged software culture (e.g., the billions of dollars to develop IBM's OS/2, Carroll, 1991; the poor quality of Ashton Tate's dBASE IV that led to the company's demise in the late 1980s; the lateness of some Microsoft products). Many packaged software projects have now become major undertakings of at least one million lines of code, increasing complexity by orders of magnitude. When complexity grows, the undisciplined American culture of software is a liability. Hackers do not have the discipline to develop large-scale high-quality software and are particularly notorious for disliking testing. Quality assurance practices have been poor (Carmel, 1993), suggesting that the "craft" approach to software development may be a way to cut corners, or worse, a form of laziness. In sum, some argue that U.S. firms practice mediocre development and have had the good fortune of an advantageous market structure.

The second criticism to the cultural thesis is that the phenomenon of the hacker culture is no longer unique to America, but has diffused to other major software nations such as Japan and Germany. These hackers appear similar to their U.S.-based peers with pop-culture artifacts that often emulate the American computer culture.

I believe that the first criticism, pointing to the negative effects of an undisciplined U.S. software culture, is a serious one. However, all cultural manifestations have negative characteristics that must be weighed against the positive ones. For U.S. packaged software firms, the positive outcomes are manifested in the successive innovation waves that have emerged from this culture. The second criticism is more superficial. While some characteristics of computer hackers have indeed globalized, other elements of the "culture of software" have not, such as the multifaceted entrepreneurial culture.

Implications for Further Sustainability

Three important influences are likely to affect packaged software competition in the years to come: multimedia, the Internet, and component-based development. However, there is little evidence that any of them will shift the U.S. home base of the packaged software industry even as some development activities are dispersing globally.

The first influence, multimedia, reinforces existing U.S. advantages of English-centered products and American hegemony in entertainment and seems unaffected by improvements in the manufacturing paradigm. Second, the Internet will continue to diffuse and democratize software development, but the important firms within the industry (e.g., Netscape and Sun Microsystems at time of this writing) continue to be U.S. based. Finally, American software developers are likely to be the key actors in the creative assembly of software components, while the more routine elements of programming will continue migrating abroad. Tom DeMarco (*IEEE Software,* 1995), an influential figure in the software community, extends this scenario by predicting that, in the long run, software will return to its roots as an invention-based development activity without any production mentality.

Notes

1. A fourth, but lesser, category is *embedded software,* so called because it is embedded in products such as VCRs, copiers, and telephones.

2. A note on conventions: When the discussion refers specifically to the *packaged* software industry, the term will be used explicitly. With broader discussion of software, the prefix "packaged" is intentionally dropped.

3. Also referred to as *commercial software* and *off-the-shelf software,* the term *packaged software* is now most common, as in *Software Magazine* (Frye & Melewski, 1995).

4. Some U.S. information technology initiatives that are noteworthy: ARPA (1970s and 1980s); MCC's Software Technology Program (1986–1991); more recently the Clinton Administration initiatives have centered on National Institute for Standards and Technology grants and the National Information Infrastructure plan.

5. Comparisons of United States and Japan in software include Zelkowitz et al. (1984) and Cusumano and Kemerer (1990).

6. A few examples of European software entrepreneurs in the United States: P. Kahn, a Frenchman, founded of one of the largest packaged software firms, Borland; two Frenchmen founded the successful U.S. firm, Neuron Data; Synon, now a top 100 software engineering firm, moved its headquarters from Britain to California.

7. Intellectual property, including copyrights and patents, has emerged as a major trade issue leading to the TRIPS Agreement (Trade-Related Aspects of Intellectual Property) within the 1994 GATT (General Agreement on Tariffs & Trade) and to a near trade war between the United States and China over intellectual property piracy in early 1995.

8. The media has also associated the term hacker with criminal or unethical acts, which, correctly, should be labeled *cracker.*

References

Alpert, M. 1992. The care and feeding of engineers. *Fortune,* 21 September, 86–95.

Brandt, R. 1991. Can the US stay ahead in software? *Business Week* March 11:98–105.

Brown, L. H., Johnson, C., and Warlick, W. 1995. *Global competitiveness of the U.S. computer software and service industries.* Washington, DC: U.S. International Trade Commission.

Business Software Alliance. 1994. *Worldwide report.* Washington, DC: BSA.

Business Week. 1994. Up front. April 11:6.

Carmel, E. 1993. How quality fits into package development. *IEEE Software* 10(5):85–86.

Carmel, E. 1995. Entrepreneurial technologists may have the upper hand in global software competition. *Res. Technol. Manage.* 38(6):10–11.

Carroll, P. B. 1991. How an IBM attempt to regain PC lead has slid into trouble. *Wall Street J.* December 2:1.

Castells, M., and Hall, P. 1994. *Technopoles of the world: The making of 21st century industrial complexes.* New York: Routledge.
Chang, C., and Burgess, A. 1991. Global issues, trends in software practice. *IEEE Software* September:4.
Clark, D. 1992. Japan still has a lot to learn about software. *San Francisco Chronicle* June 15:B1.
Cottrell, T. 1994. Fragmented standards and the development of Japan's microcomputer software industry. *Res. Policy* 23(2):143–174.
Cringely, R. X. 1992. *Accidental empires.* Reading, MA: Addison-Wesley.
Cusumano, M. A. 1991. *Japan's software factories: A challenge to US management.* New York: Oxford University Press.
Cusumano, M. A., and Kemerer, C. F. 1990. A quantitative analysis of US and Japanese practice and performance in software development. *Manage. Sci.* 36(11):1384–1406.
Cusumano, M. A., and Selby, R. W. 1995. *Microsoft secrets.* New York: Free Press.
Datamation 100. 1993. *Datamation* 39(12):22.
Dedrick, J. L., Goodman, S. E., and Kraemer, K. L. 1995. Little engines that could: Computing in small energetic countries. *Commun. ACM* 38(5):21–26.
Derian, J. 1990. *America's struggle for leadership in technology.* Cambridge, MA: MIT Press.
Dvorak, J. C. 1993. Inside track. *PC Magazine* October 12:95.
Economist. 1994. Europe's software débâcle. November 12:77–78.
Fox, R. 1995. Newstrack. *Commun. ACM* 38(7):9.
Frye, C., and Melewski, D. 1995. Special report: Top 100. *Software Mag.* 15(7).
Gibbs, W. W. 1994. Software's chronic crisis. *Sci. Am.* September 271:85–95.
Greenbaum, J., and Johnston, M. 1993. Euro-entrepreneurs in a bind. *Upside* 5(10):65–82.
Hamilton, D. P. 1993. U.S. companies rush to fill Japanese software gap. *Wall Street J.* May 7:B4.
Hofstede, G. 1980. *Culture's consequences: International differences in work-related values.* Beverly Hills, CA: Sage.
Hofstede, G. 1993. Cultural constraints in management theories. *Acad. Manage. Exec.* 7(1):73–94.
Hudson, R. L. 1994. European software makers face struggle, *Wall Street J.* July 22:A8.
IEEE Software. 1995. Interview with Tom DeMarco. March:108–109.
Jones, T. C. 1993. *Software productivity and quality today: The worldwide perspective.* Carlsbad, CA: IS Management Group.
Jones, T. C. 1994a. Can the US stay on top? *Information Week* January 10:47.
Jones, T. C. 1994b. Globalization of software supply and demand. *IEEE Software* 11(6):17–24.
Koertz, G. 1995. Computers may be really paying off. *Business Week.* February 14.
Kogut, B. 1991. Country capabilities and the permeability of borders. *Strategic Manage. J.* 12:33–47.
Kosnick, T. 1990. Microsoft Corporation: The Introduction of Microsoft Works. Harvard Business School Teaching Note 5-590-076. Cambridge, MA: Harvard Business School.
Kyle, J. 1994. To hack is human, to engineer divine. *Am. Programmer* January:9–13.
Lammers, S. 1986. *Programmers at work.* Redmond, WA: Microsoft Press.
Levering, R., Katz, M., and Moskowitz, M. 1984. *The computer entrepreneurs.* New York: New American Library.
Mansfield, E. 1988. Industrial R&D in Japan and the United States: A comparative study. *Am. Econ. Rev.* 78(2):223–228.
Mori, K., and Kawada, T. 1990. From kana to kanji: Word processing in Japan. *IEEE Spectrum* August:46–48.
Neff, R. 1995. Show Biz: Don't count Japan out. *Business Week* April 24:126.
Nidumolu, S. R., and Goodman, S. E. 1993. Computing in India: An Asian elephant learning to dance. *Commun. ACM* 36(6):15–22.
Pacey, A. 1983. *The culture of technology.* Cambridge, MA: MIT Press.
Peters, T. 1995. Do it now, stupid. *Forbes* August 28:172.
Phillips, D. 1994. Building a cultural index to world airline safety. *Washington Post* August 21:A8.
Porter, M. E. 1990. *The competitive advantage of nations.* New York: Free Press.

Press, L. 1993. Software export from developing countries. *Computer* 26(12):62–67.
Price Waterhouse. 1993. Software Industry Business Practices Survey. Boston: Price Waterhouse.
Quintas, P. 1994. Ivory tower power. *London Times,* Higher Education Supplement, Multimedia section, February 4:6.
Ravichandran, R., and Ahmed, N. 1993. Offshore systems development. *Information Manage.* 24(1):33–40.
Reich, R. B. 1987. Entrepreneurship reconsidered: The team as hero. *Harvard Business Rev.* 65(3):77–83.
Roberts, E. B. 1991. *Entrepreneurs in high technology: Lessons from MIT and beyond.* New York: Oxford University Press.
Sakakibara, K., and Westney, D. E. 1992. Japan's management of global innovation: Technology management crossing borders. In *Technology and the wealth of nations,* eds. N. Rosenberg, R. Landau, and D. Mowery. Stanford, CA: Stanford University Press.
Scherer, F. M. 1992. *International high-technology competition.* Cambridge, MA: Harvard University Press.
Schrage, M. 1993. Software powers remain elusive for Japanese. *Los Angeles Times* May 13:D1.
Schware, R. 1989. The world software industry and software engineering. World Bank tech. paper no. 104. Washington, DC.
Schware, R. 1992. Software industry strategies for developing countries: A walking on two legs proposition. *World Dev.* 20(2):143–164.
Schwartz, E. I. 1992. Twilight of the nerds? Programming must go beyond grunt work. *Business Week* October 23:174.
Siwek, S. E., and Furchtgott-Roth, H. W. 1993. *International trade in computer software.* Westport, CT: Quorum Books.
Siwek, S. E., and Furchtgott-Roth, H. W. 1995. Copyright Industries in the U.S. Economy: 1977–1993. Washington, DC: Economists Incorporated, January.
Software Magazine. 1994. Software fact. October:22.
Taylor, P., and Guptara, P. 1995. India's software industry. *Financial Times,* special report, December 6.
Upside. 1995. 100 Hot Private Technology Companies. pp. 26–34. June.
U.S. Department of Commerce. 1995. U.S. Global Trade Outlook 1995–2000. Washington, DC.
von Hippel, E. 1988. *The sources of innovation.* New York: Oxford University Press.
Watanabe, T. 1990. Japan's trying hard to catch U.S. in software. *Los Angeles Times* July 8:D1.
Yourdon, E. 1992. *The decline and fall of the American programmer.* Englewood Cliffs, NJ: Yourdon Press.
Zachary, G. P. 1994. *Showstopper: The breakneck race to create Windows-NT and the next generation at Microsoft.* New York: Free Press.
Zelkowitz, M. V., Yeh, R. T., and Hamlet, R. G. 1984. Software engineering practices in the US and Japan. *IEEE Computer* 17:57–66.
Zuconni, L. 1990. U.S. technology: We're losing the edge in software. *J. Systems Software* 12:53–58.