MANAGEMENT SCIENCE
Vol 36, No. 11, November 1990
Printed in U.S.A

A QUANTITATIVE ANALYSIS OF U.S. AND JAPANESE PRACTICE AND PERFORMANCE IN SOFTWARE DEVELOPMENT*

MICHAEL A. CUSUMANO AND CHRIS F. KEMERER
Massachusetts Institute of Technology, Sloan School of Management, Cambridge, Massachusetts 02139

Since the early 1980s, there has been a mounting debate in industry literature and in U.S. government-sponsored reports over the relative performance of software developers in Japan versus those in the United States. This literature is somewhat divided between assertions of Japanese or U.S. superiority in this technology, although many observers in the popular business press continue to insist that the U.S. maintains an overwhelming lead in this technology. However, both sides of the debate have offered evidence that, to date, has been primarily qualitative or based on one or two cases.

This paper contributes to the debate in two ways. First, it offers a comprehensive literature review that analyzes existing comparisons of Japanese and U.S. practice in software development and summarizes the major proposed differences in performance. Second, it presents the first set of quantitative data collected from a statistically comparable sample of 24 U.S. and 16 Japanese software-development projects, and uses these data to test propositions from the literature. The analyses indicate that Japanese software projects perform at least as well as their U.S. counterparts in basic measures of productivity, quality (defects), and reuse of software code. The data also make it possible to offer models that explain some of the differences in productivity and quality among projects in both countries.

(SOFTWARE DEVELOPMENT PRODUCTIVITY; SOFTWARE QUALITY; SOFTWARE REUSE; JAPANESE SOFTWARE; JAPANESE MANAGEMENT METHODS)

Introduction

Japanese firms have become well known for transferring and adapting imported technology, initially concentrating on commodity product designs, innovating in product engineering and manufacturing to achieve high productivity and quality, and then gradually increasing the sophistication and variety of their products. This approach has worked effectively in industries such as steel, ships, automobiles, consumer electronics, machine tools, semiconductors, and computer hardware. Yet, despite this impressive record of successes, some observers of Japan and academic researchers have argued that this strategy will not work with new, still-evolving technologies where customers and producers have yet to define product or process standards, and where product development requires considerable creativity or individual innovation. A major challenge for the Japanese—and the subject of this paper—would thus seem to be the development of computer software, one of the few industries where U.S. firms, accounting for as much as 75% of worldwide sales in the early 1980s, continued to dominate Japanese competitors in international markets (U.S. Department of Commerce 1984, p. 33).

Software has presented problems to managers since the beginning of the industry in the late 1950s, when programmable computers first appeared. The development process consists of various levels of system design, coding, and testing, as well as redesign or repairs referred to as maintenance. Yet software producers continue to experience cost and schedule overruns as the rule rather than the exception, especially since huge variations in customer requirements, frequent design changes during development, and wide discrepancies in individual productivity have complicated managerial tasks. Characteristics

* Accepted by Alok K. Chakrabarti; received August 21, 1989. This paper has been with the authors 2 months for 1 revision.

0025-1909/90/3611/1384$01.25

of the industry and the technology thus produced a situation referred to as the "software crisis" as long ago as 1969 (Hunke 1981, Frank 1983), and problems cited in the 1960s continued to plague software producers in the 1980s (Arden 1980, Boehm 1987, Ramamoorthy 1984, Brooks 1987). Meanwhile, industry analysts, academics, and practitioners have argued over whether it is even possible to manage software as a predictable engineering and manufacturing discipline, as opposed to an art or craft, highly dependent on the variable skills and experience of particular individuals (Brooks 1975, Shooman 1983, Hauptman 1986).

This study compares Japanese and U.S. project performance in software development and offers exploratory but quantitative data on a debate that, up to the present, has proceeded largely in an anecdotal and qualitative fashion. The disagreements center around whether U.S. firms continue to dominate Japanese competitors in software development or whether Japanese firms have been able to apply basic skills in engineering and production management to achieve comparable or even higher levels in key areas of performance in software development, such as productivity, defects, and reusability of code. To examine this debate, the first section of this study reviews existing literature to identify statements observers have made regarding the Japanese and U.S. in software development. The second section explains the research methodology followed to confirm or deny several propositions indicated in the literature. This methodology consisted of the collection of an exploratory set of quantitative data on a sample of actual software projects done in the U.S. and Japan. The subsequent sections analyze the sample in some detail, including descriptive information, process data and performance comparisons, and a sensitivity analysis of the sample data. The analysis uses project information primarily to examine propositions stated in the literature and then compare Japanese and U.S. projects, although it also attempts to elicit some of the causes underlying differences in productivity and quality among different software development projects in general. The concluding section summarizes and interprets the results and then suggests implications for future research.

1. Literature Review: Propositions in the Debate

A review of more than a dozen sources published between 1969 and 1989 revealed a debate with two sides, neither well supported. On the one hand were a rising number of claims that Japanese projects actually performed well in the *process* of software development: high levels of productivity, quality, tool usage, and reusability, as well as discipline, teamwork, and planning. Most of the literature offered general comments, for example, that problems common in programming might actually benefit from skills Japanese firms and workers seem to possess in abundance, such as excellence in planning, problem-solving, process management and attention to detail, willingness to cooperate and communicate, high motivation among workers and managers, creativity in nearly all things mechanical, and general perseverance.[1]

A few publications also cited some limited quantitative data. For example, a 1983 survey of 10 Japanese software producers and R&D organizations, sponsored in part by the U.S. Office of Naval Research, described Toshiba's Software Factory as "one of the most advanced real-time software engineering organizations in the world," with productivity averaging 2870 instructions per programmer month in 1981, due to the use of integrated tool sets (Kim 1983). A 1984 U.S. government study asserted that, "Japanese programmers average 2000 lines of code per month (versus less than 300 lines per month

[1] Software engineering researchers who make some of these claims are Zelkowitz et al. 1984 and Belady 1986. In addition, although the literature on Japanese management styles and employee behavior is too long to cite comprehensively, examples for interested readers include Vogel 1979, Cole 1979, Schonberger 1982, and Abegglen and Stalk 1985.

for U.S. programmers) and have one-tenth the error rate (defects) of their U.S. counterparts" (U.S. Dept. of Commerce, p. 11). A 1986 article discussing Toshiba also cited a 65% reuse rate for delivered code, productivity of 2000 lines per month per programmer, and merely 0.3 defects per 1000 lines of code (Haavind 1986). In the *product* area, observers disagreed more widely over Japanese performance, although some felt the Japanese might be at least comparable to U.S. counterparts in custom applications programming for the Japanese market and specific software areas (real-time applications, graphics and video programs, super-computer programs, on-line reservation systems, and embedded software in consumer and industrial products) (Table 1).

On the other side of the debate, various observers concluded that, while the Japanese had made progress in productivity, quality, and project management, they still lagged

TABLE 1
Positive Comments About Japanese Software Development

Comments:	Sources:	Quantitative Data:
	PROCESS	
Superior Project Management (Planning, Discipline, Teamwork)	Naur and Randall 1969	None
	Tajima & Matsubara 1981	1 Company's Data
	Zelkowitz et al. 1984	None
	U.S. Commerce 1984	None
	Johnson 1985	None
	Belady 1986	None
More Code Reuse	Standish 1984	Anecdotal
	Haavind 1986	Anecdotal
	Gamaota & Frieman 1988	1 Company's Data
	Tracz 1988	Anecdotal
	Cusumano 1989	Manager Survey
More Tool Usage	Kim 1983	None
	Zelkowitz et al. 1984	Site Survey
	U.S. Commerce 1984	None
	Johnson 1985	None
	Gamaota & Frieman 1988	None
Higher Productivity	Kim 1983	1 Company's Data
	U.S. Commerce 1984	Anecdotal
	Haavind 1986	1 Company's Data
	Gamaota & Frieman 1988	1 Company's Data
Better Quality (Fewer Defects) and Product Engineering	Zelkowitz et al. 1984	None
	Johnson 1985	None
	U.S. Commerce 1984	None
	Haavind 1986	1 Company's Data
	OTA 1987	1 Company's Data
	Gamaota & Frieman 1988	1 Company's Data
	PRODUCTS	
Good in Custom Programming	Cusumano 1988	User Surveys
	OTA 1987	None
Good in Specific Applications (Real-Time, AI, Graphics, Supercomputer, MIS, On-Line Reservations, Embedded, Jap. language processing)	Kim 1983	None
	Uttal 1984	None
	Sakai 1984	None
	Gamaota & Frieman 1988	None
	Lecht 1989	None

significantly behind the U.S. overall in the software field. The arguments ranged from statements that Japanese projects relied on tools and techniques adopted from the U.S. and Europe to accounts of the small size of the Japanese software industry and Japan's continuing and severe shortage of skilled programmers. To outside observers, Japanese engineers also seemed to lack creativity and the ability to invent new or sophisticated products, especially software packages. In apparent contradiction to accounts of very high productivity in Japan, other observers believed Japanese software was more costly to develop than U.S. software, suggesting low development productivity or inefficiency in other areas. In addition, nearly all reports found the Japanese behind in basic research and advanced product development. These and other reasons made it seem unlikely Japanese firms could compete effectively in the world software market (Table 2). As noted in Table 2, for example, a 1987 study by the U.S. Office of Technology Assessment (OTA) concluded that "Japan remains substantially behind in software, with poor applications packages—along with limited sales and service networks . . ." (Office of Technology Assessment 1987). Two articles from 1989 argued that the Japanese software industry remained "small and not very visible," leaving the Japanese "two decades" behind the U.S. in software development (Rifkin and Savage 1989), with Japan suffering from a severe shortage of skilled programmers and few software houses strong enough to compete with U.S. firms in the American market (Lecht 1989).

TABLE 2

Critical Comments About Japanese Software Development

Comments:	Sources:	Quantitative Data:
	PROCESS	
Copy/Rely on Western Tools and Techniques	Kim 1983	None
	Zelkowitz et al. 1984	None
	Kishida 1987	None
	Cusumano 1989	Managers' Survey
Severe Shortage of Skilled Programmers	Lecht 1989	None
More Costly Development	Uttal 1984	None
	PRODUCTS	
Lack Creativity	Uttal 1984	None
	Rifkin and Savage 1989	None
No Inventions	U.S. Commerce 1984	Historical Lists
Less Sophisticated	Uttal 1984	None
Few Packages	Kim 1983	None
	U.S. Commerce 1984	None
	OTA 1987	Industry Data
Behind in Basic Research and Advanced Development	U.S. Commerce 1984	None
	Gamaota & Frieman 1988	Anecdotal
	GENERAL ASSESSMENTS	
Behind the U.S. Overall	U.S. Commerce 1984	None
	Sakai 1984	None
	OTA 1987	Industry Data
	Rifkin and Savage 1989	None
	Lecht 1989	None
	Lewis 1989	None

For outside analysts and researchers, a frustrating aspect of this debate has been the conflicting statements and their anecdotal nature—general comments that are difficult to test, such as assertions that Japanese products lacked sophistication or creativity; and claims that relied on information from one or two firms or projects, with little or no attempt to collect quantitative data systematically. Part of the reason is the difficulty of measuring performance in software programming or product operation, especially across different users, producers, and projects (Jones 1986), as well as national markets. Nevertheless, the reports cited in Tables 1 and 2 contained specific claims that appear suitable for quantitative analysis. Although systematic product analyses were beyond the scope of this study, if U.S. software projects were to emerge as clearly superior in basic measures of process performance, this finding would lend support to arguments that the Japanese continue to lag behind the U.S. and that the type of skills in conventional engineering and production management the Japanese have displayed in other industries may not be suitable for software development. On the other hand, strong results from Japan would lend support to claims that U.S. firms no longer have a clear advantage in this technology and that skills or practices useful for effective management of software development may indeed be similar to those in other fields. In either case, project analysis of the sort pursued in this research should serve as a basis for additional empirical and theoretical exploration of variables related to the effective management of software development in general. Accordingly, an analysis of literature cited in Tables 1 and 2 reveal the following testable propositions:

Descriptive Data

PROPOSITION 1. *Japanese software projects use similar languages and hardware as in U.S. projects* (*Table* 2).

PROPOSITION 2. *Japanese software projects develop systems that are less complex or sophisticated than in U.S. projects* (*Table* 2).

Process Data

PROPOSITION 3. *Japanese software projects exhibit superior management characteristics compared to U.S. projects* (*Table* 1).

PROPOSITION 4. *Japanese software projects reuse more code than U.S. projects* (*Table* 1).

PROPOSITION 5. *Japanese software projects use more tools than U.S. projects* (*Table* 1).

Performance Data

PROPOSITION 6. *Japanese software projects exhibit higher lines-of-code productivity than U.S. projects* (*Table* 1).

PROPOSITION 7. *Japanese software projects exhibit fewer defects per lines of code than U.S. projects* (*Table* 1).

2. Research Methodology

The methodology adopted to compare Japanese and U.S. practice and performance in software involved the collection of data on standardized forms from individual projects building identifiable systems. The first task compiled a list of major software producers and development sites in each country that appeared comparable in terms of system or project size and applications. Names of firms came from annual lists of the largest software producers in U.S. and Japanese publications (*Datamation* 1985, Kiriu 1986). Annual

reports and other company information indicated the location of major software sites. The next step identified managers of software development at these sites willing to complete a standardized data-collection form on one or more projects of their choice. The approach taken was to contact the managers of production engineering or quality assurance in each development site by letter and by telephone, explain the purpose of the study and ask for their cooperation in return for a copy of the research report.

Managers at 44 development sites in the United States (and one from a Japanese joint venture returned from its U.S. partner) initially agreed to complete the form and submit data on projects of their choice; 28 were returned, from 12 firms. Managers at 26 development sites in Japan also agreed to complete the form on projects of their choice; 20 were returned, from 9 firms (see Appendices A and B). After review of the returned forms, augmentation of incomplete forms or clarification of responses was attempted by telephone or letter. The end result of this additional effort was that four of the U.S. projects and four of the Japanese were not usable. Either they did not include enough data or the respondent who filled out the survey indicated too many doubts about the reliability of the data. In addition, the one returned by a U.S. company from its joint venture in Japan was excluded from the study, since this appeared to represent a combination of Japanese and U.S. practices and personnel. Thus the response rate for returns was 48 out of 70 (69%), and 40 out of the 48 projects (83%) were used in this research, although not every data-collection form contained complete information on each question.[2]

The high percentage of returns (given the detailed data required) seemed due to the personal commitments the researchers sought from individuals in each firm and continued appeals by telephone and letters until most of the surveys sent out came back completed. Companies were also promised confidentiality, i.e., no project data would be directly associated with a particular company. One obstacle to getting a larger sample was that, for the U.S. projects in particular, many managers did not collect data (such as on work years by phase of development, lines of code produced, defect levels by degree of severity and over time) in sufficient detail to participate. This contrasted to the Japanese, where data collection appeared to be more routine and thorough. Also, some managers in both countries, even after initially agreeing to submit data, elected not to divulge this information on productivity and quality for competitive reasons, even when promised confidentiality.

It should be noted that the sample was not random. The research identified leading producers who collected detailed information on their development processes. The research also followed the methodology used by Zelkowitz et al. (1984), allowing managers to select on what projects to report data. One must therefore assume that companies probably chose their best projects, or at least projects under sufficient control for them to have fairly detailed data on productivity, quality, and other measures. In short, the sample remains exploratory, but it should give an indication of "good practice" in the U.S. and Japan, at a selected group of firms that collected detailed data and took an interest in the management of software development. It also provides a unique opportunity to assess *quantitatively* the claims and counter claims raised by anecdotal reports over the last decade as a precursor to further research on factors affecting software project management in any setting.

3. Descriptive Data

This section compares the Japanese and U.S. samples across several basic dimensions: application type, programming language, hardware platform, and system size. The data-

[2] For example, 9 of the 40 did not report quality (defect) data.

TABLE 3
Applications Developed

Applications	U.S. (%)	Japan (%)
Data Processing	8 (33)	2 (13)
Scientific	1 (4)	3 (19)
Systems	4 (17)	6 (38)
Telecomm./Realtime	11 (46)	5 (31)
	24	16

collection forms, in addition to requesting information on productivity and quality, requested a brief description of the purpose of the software. This allowed the researchers to characterize the 40 software systems into one of the standard applications described by Jones (1986): (1) data processing (e.g., financial database, human resource), (2) scientific (e.g., simulation models, CAD tool), (3) systems software (e.g., operating system, compiler), and (4) telecommunications and other real-time systems (e.g., switching, data transmission, and network processing). The distribution of projects in the total sample, shown in Table 3, appears comparable although not identical. To determine if product type affected performance measures, the analyses that follow include variables to represent possible differences in application mix across the two samples.

PROPOSITION 1. *Japanese software projects use similar languages and hardware as in U.S. projects* (*Table* 2).

Lists of applications, programming languages, and hardware platforms, as well as the tools used in software development, indicate that Japanese projects have largely followed the lead of the U.S., where this technology was invented. For example, primary application languages are quite consistent across the two countries (see Table 4a). The U.S. projects have somewhat more representation in Assembly and COBOL, although this reflected the greater percentage of real-time and data-processing systems, respectively (see Table 4b). The percentages of types of hardware platforms on which they were delivered are quite similar, even though the Japanese projects have a slightly greater percentage of microcomputer implementations, and the U.S. sample a small advantage in mainframes (Table 5). Therefore, in terms of applications, languages and hardware platforms, there do not appear to be great differences between the U.S. and Japanese projects in this sample.

PROPOSITION 2. *Japanese software projects develop systems that are less complex or sophisticated than in U.S. projects* (*Table* 2).

The terms "complex" and "sophisticated" are vague in general, especially in the context of software. One aspect might be the type and number of functions available in a program, although neither this study nor any other reviewed here has directly compared Japanese

TABLE 4a
Primary Programming Languages

Primary Language	U.S. (%)	Japan (%)
Assembly	5 (21)	2 (13)
C	3 (13)	3 (19)
Cobol	6 (25)	2 (13)
Fortran	3 (13)	2 (13)
PL/1	2 (8)	4 (25)
Pascal	1 (4)	0 (0)
Other	4 (17)	3 (19)
	24	16

TABLE 4b
Primary Programming Language by Application

	Data Processing	Scientific	Systems	Telecommunications and Other Real-Time
Cobol	8	—	—	—
Pascal	1	—	—	—
Fortran	1	2	1	1
C	—	2	2	3
PL/1	—	—	3	3
Assembly	—	—	3	4
Other	—	—	1	5
Total	10	4	10	16

and U.S. software in terms of functionality. The writers who claimed Japanese software lacked sophistication specifically stated or implied that Japanese programs appeared less "complex" than U.S. software (Uttal 1984, OTA 1987). If one accepts that there may be an association between the number of functions and the size of a system, then the data collected make it possible to compare the Japanese and U.S. systems in one area, size, that is an aspect of sophistication or complexity.

The conventional measure for size in a software project is the number of noncomment source lines of code (SLOC) produced (Boehm 1981, Conte et al. 1986, Jones 1986, Putnam 1978, Walston and Felix 1977). This metric has a long history in both research and practice, and has been the subject of much debate concerning rules for counting SLOC and their efficacy as an input metric for project estimation as well as a measure of productivity (Albrecht and Gaffney 1983, Jones 1986, Kemerer 1987). The current state of the debate is best summarized in a recent article by a founder of the software-engineering field, Barry Boehm: "The current bottom line for most organizations is that delivered source instructions (lines) per project man-month . . . is a more practical productivity metric than the currently available alternatives" (Boehm 1987).

In order to compare system size as well as productivity across a number of organizations, this research chose noncomment SLOC as the output size metric and work-years as the input size metric.[3] Table 6 shows the means and medians of these metrics. Note that the U.S. and Japanese systems are of roughly comparable size, and the results of a nonparametric Wilcoxon rank sum test (equivalent to a Mann-Whitney U test) indicated no statistically significant difference (Bradley 1968, pp. 105–114).

TABLE 5
Hardware Platforms[a]

Hardware Platform	U.S. (%)	Japan (%)
Mainframe	11 (52)	7 (47)
Minicomputer	6 (29)	4 (27)
Microcomputer	4 (19)	4 (27)
	21	15

[a] Note that only 36 of the 40 systems are counted as systems with multiple platforms are excluded.

[3] One possible concern about such a measure is the reputation of Japanese firms for long work days. For example, in the automobile industry, Japanese employees tended to work about 15% more days per year than their U.S. counterparts (Cusumano 1985). However, it appears that hourly differences were relatively minor for U.S. and Japanese software developers. In Japan, 1987 figures indicate the average weekly hours for the information-processing services industry were 37.6 and, for the software industry (system engineering, programming, operations), 37.3 hours. The Japanese reported modest overtime hours per week, averaging 6.2 in information-processing services and 7.7 in the software industry (Joho Sabisu Sangyo Kyokai 1987, pp. 118–

TABLE 6
Input and Output Metrics

Size	Means U.S. (median)	Means Japan (median)	Wilcoxon Rank Sums Z Approximation
Work-years	102 (22.5)	47 (20.1)	0.00
SLOC	343 K (124 K)	433 K (164 K)	0.54
Fortran Equivalent	288 K (77 K)	389 K (144 K)	0.62
Average Fortran Conversion	0.90	0.94	0.38

One possible concern with the use of SLOC metrics for interproject comparisons is the variance in programming languages. "A line" of Fortran may not be equivalent to "a line" in Assembly language, for example. To control for this source of variance, the SLOC measures were converted to a common size of Fortran equivalent statements, using the conversion factors proposed by Jones (1986, p. 49, and 1988). For example, using languages in the current sample, Assembly language is at "level" 1, and Fortran is at "level" 3. It thus takes 3 lines of Assembly to equal 1 line of Fortran. This conversion was performed on all of the SLOC data, and the results are also shown in Table 6. Note that the conversion produces Fortran-equivalent LOC that are slightly smaller than the raw SLOC data, due to the languages used. Note also that the relative numbers between the U.S. and Japanese companies do not change significantly, which would be expected given the similar language set in each country's data. It follows that no support is found for Proposition 2, that Japanese software projects develop less complex or sophisticated systems than in the U.S., at least to the degree that system size reflects complexity and sophistication.

4. Process Data

In addition to the descriptive data described in §3, data were collected on the process of software development as practiced in the U.S. and Japanese projects. These data consist of both the labor (project staffing) and capital inputs (reused code and tools).

TABLE 7
Personnel Experience

	Average Years of Experience U.S.	Average Years of Experience Japan	Wilcoxon Rank Sums Z Approximation
Programmer	3.48	3.31	−0.94
Designer	4.46	4.44	0.35
Manager	7.61	7.00	0.00

119, 133). In the U.S., for SIC code #737 (computer programming, data processing, and other computer-related services), average regular hours worked per week (excluding overtime) in 1987 were 37.5, nearly identical to Japan. U.S. employees also worked overtime as needed, although the U.S. Department of Labor did not collect these data for service industries (U.S. Department of Labor 1989).

TABLE 8
Effort by Phase

	Effort Distribution by Phase		Wilcoxon Rank Sums Z Approximation
	U.S.	Japan	
Design %	31	39	2.71***
Coding %	36	25	1.97**
Testing %	33	36	0.66

Statistical Significance Levels: ** = 0.05, *** = 0.01.

A. *Labor Inputs*

One widely used surrogate for the quality of personnel employed on a project is their average years of experience (Chrysler 1978, Banker et al. 1987). As can be seen in Table 7, these levels are essentially identical for the data from the two countries. This contrasts with claims (such as Lecht 1989) of greater shortages of experienced software developers in Japan compared to the U.S. Therefore, any existing management or performance differences between the countries in this sample are unlikely to be explained by differences in experience as measured by years of employment in software development.

PROPOSITION 3. *Japanese software projects exhibit superior management characteristics compared to U.S. projects* (*Table* 1).

Previous researchers have suggested that performance differences may be due to increased emphasis on the initial stages of the development life cycle (Gaffney 1982, McKeen 1983). In order to test the claims of superior Japanese project management, data were collected on how projects used personnel, in particular on the allocation of effort across the systems development life-cycle.

Average data relating to life-cycle emphasis as collected on the individual projects are shown in Table 8. For purposes of this data-collection effort, "Design" includes the pre-coding specification as well as the design phase. "Coding" includes programming, and "Testing" includes debugging. As can be seen, on average the Japanese spend significantly more (at the alpha = 0.01 level) time in the early life-cycle phase, and significantly less (at the alpha = 0.05 level) in the coding phases. These data provide support for Proposition 3 to the degree that emphasizing design and de-emphasizing coding is seen as desirable. The difference in testing percentage was also higher, although this difference was not statistically significant at usual levels.

Data were also collected on the composition of the work-years in terms of full-time versus part-time project participation in each of the phases. As shown in Table 9, these

TABLE 9
Full-Time Effort Percentage by Phase

	Full-Time vs. Part-Time %		Wilcoxon Rank Sums Z Approximation
	U.S.	Japan	
Full time design	82	77	−0.38
Full time coding	81	81	0.07
Full time testing	61	86	1.95**

Statistical Significance Levels: ** = 0.05.

TABLE 10
Code Reuse

	Mean Code Reuse (% of Delivered Lines)		Wilcoxon Rank Sums Z Approximation
	U.S. (median)	Japan (median)	
Code reuse	9.71 (3)	18.25 (11)	0.71

data are similar for the two countries in the design and coding phases, but the Japanese projects show significantly greater reliance on full-time testing personnel than do the U.S. projects. This is another difference in project management that supports suggestions by other researchers that Japanese firms excel in defect analysis (Zelkowitz et al. 1984) and other aspects of quality control (see Table 1).

B. *Capital Inputs*

A number of researchers have used an economic production process model for software development to suggest that one way to improve productivity is to emulate industries that substituted reliable, low marginal-cost capital inputs for repetitive activities (Kriebel and Raviv 1980, Stabell 1982, Kemerer 1988, Cusumano 1989). In the domain of software, capital inputs most readily take the form of tools and techniques for augmenting human labor. For this study, data were collected on two forms of capital input, code reuse and software tools.

PROPOSITION 4. *Japanese software projects reuse more code than U.S. projects (Table 1).*

The issue of code reuse has received much attention recently as a potential "silver bullet" for the software crisis (Brooks 1987).[4] Unfortunately, with only a few exceptions (e.g., Selby 1989), little empirical data has been published on actual reuse in industrial settings. The data for the U.S. and Japanese projects are shown in Table 10.

Consistent with previous reports (see Table 1), the Japanese projects seem to exhibit higher levels of reuse than U.S. projects. Higher reusability in Japan would also be consistent with assertions that the Japanese have taken the lead in promoting software reuse, even without solving all the accompanying difficulties (Tracz 1988). However, the difference in this sample is not statistically significant at usual levels. Therefore, while the data appear to support Proposition 4, they cannot be used to reject the possibility there may be no difference in reuse levels between Japanese and U.S. projects. The data also suggest that the very high percentages of reuse for the Japanese cited by some researchers (such as 85% in Standish 1984) represent isolated best practice or unusual projects.

PROPOSITION 5. *Japanese software projects use more tools than U.S. projects (Table 1).*

Another area of software development technology that has received considerable research attention is the provision of tools, particularly so-called CASE (Computer-Aided Software Engineering) tools, to support software development (Henderson and Cooprider 1988). In order to investigate their degree of use in the U.S. and Japan, data were collected with an open format design, as follows:

[4] See also the collection of papers in Freeman 1987, and the special issue of *IEEE Transactions on Software Engineering*, September 1984.

Software Engineering Tools: List in importance the name and main function of the most frequently used support and development tools for product design, coding and testing.

	Name	*Main Function*
1.		
2.		
. . .		
10.		

An alternative would have been to use a closed format, asking respondents to check the applicable boxes. It is believed that this area of practice is too ill established for such an approach. The open format at least permitted an exploratory evaluation of the scope and breadth of tool usage. Table 11 shows the average number of tools listed by projects in each of the two countries.[5] This gross level of analysis suggests that Japanese and U.S. projects had similar levels of tool usage.

Due to the detailed level of the data, a finer level of analysis can be performed, as shown in Table 12, which presents the tool-usage statistics broken down by type of tool used. By decomposing the usage into these categories, the number of projects using any given type of tool become very small, and therefore standard statistical tests seem inappropriate. Examination of the table indicates that these Japanese and U.S. projects used a comparable range of tools, but with a few differences. In terms of analysis and design, similar numbers of projects reported using tools to support these activities, although the Japanese reported greater use of automatic flow-charting tools. In coding, Japanese projects reported much greater use of what are referred to in Table 12 as "utilities," e.g., such tools as domain-specific editors and specialized compilers. Also, none of the U.S. projects reported using either code generators or reusable code libraries, which are tools developed in the U.S. and promoted by U.S. experts in software engineering (Boehm 1981, Brooks 1987).

Note that zero use of code libraries is not necessarily inconsistent with the code reuse numbers cited earlier. Studies have described how code reuse can be done formally, through corporate or department code libraries, or informally, through ad hoc reuse and private libraries (Woodfield et al. 1987). This is an important distinction for managers, since there is some debate in the reuse literature regarding the cost and benefits of reuse, and the variance in how reuse is supported institutionally has been suggested as a factor that has an impact on these costs and benefits (Matsumoto 1987, Cusumano 1989).

While data on testing or debugging tools is similar across the U.S. and Japanese projects, the percentage of respondents claiming use of these tools was relatively high (roughly two-thirds), compared to data cited by Zelkowitz et al. (1984), which shows only 27% claiming use. The difference in these two observations may stem from the five-year dif-

TABLE 11
Tool Usage

	Mean Tools/Methods Reported Used Per Project		Wilcoxon Rank Sums *Z* Approximation
	U.S. (median)	Japan (median)	
Number Used	4.04 (4)	4.13 (4)	−0.06

[5] The form permitted a maximum of 10 responses, and two of the 40 responses (one from each country) actually included 10 tools. Therefore, for these two data points a possible methods bias exists in terms of a ceiling effect. However, given that this was the case for only 2 of the 40 responses and that there is one from each country, this is not believed to be a major source of error.

TABLE 12
Detailed Tool Usage

	Types of Tools Used—Percentage (%) Reporting Use	
	U.S.	Japan
Analysis/Design		
Design support	29.2	31.3
Auto. Flow-chart	4.2	18.8
Coding		
Utilities	29.2	75.0
Envir. Mgmt.	37.5	25.0
Data Mgmt.	12.5	6.3
Code Generators	0.0	6.3
Reuse/Pgm. Lib.	0.0	18.8
Testing		
Test/Debug	66.7	62.5
Simulators	25.0	18.8
Performance Testing	4.2	0.0
Other Tools		
Docum. Support	8.3	6.3
Schedule Tracking	16.7	0.0
Problem Tracking	29.2	6.3
Metrics Collection	12.5	12.5
Miscellaneous	16.7	18.8

ference in when the two sets of data were collected, which would suggest that usage of such tools has become much more common. Other tools showed similar levels of use across the two countries. Thus, in general, the data do not lend support to Proposition 5. The Japanese projects did claim far more extensive use of coding utilities and used automated flow-charting and reuse-support tools that the U.S. projects did not, although the U.S. projects listed some tools (performance testing, schedule tracking) that the Japanese did not.

5. Performance Metrics and Models

The literature suggests that the relative performance of U.S. and Japanese software developers has become of great interest to managers concerned with identifying and understanding good practice in the software industry as well as to U.S. government analysts and others concerned with the international competitiveness of U.S. firms. As noted earlier, numerous reports still conclude the Japanese remain behind the U.S. in software development and are unlikely to pose a threat to U.S. firms in this industry (Table 2). This section presents the results of a quantitative analysis of productivity and quality based on the current sample.

PROPOSITION 6. *Japanese software projects exhibit higher lines-of-code productivity than U.S. projects* (*Table* 1).

Productivity is defined here as a noncomment Fortran-equivalent source lines of code per work year (Jones 1986, 1988). Table 13 presents the comparison of the U.S. and

TABLE 13
Mean Productivity (Fortran-Equivalent SLOC/Work-Year)

	U.S. (median)	Japan (median)	Wilcoxon Rank Sum Z Approximation
Fortran productivity	7290 (2943)	12447 (4663)	1.34

Japanese projects along this dimension.[6] Both the mean and median for the Japanese are higher (58% to 71%) than the U.S. numbers, although the differences are not statistically significant at usual levels (alpha = 0.18). Therefore, the data, while seeming to support Proposition 6, cannot be used with a high degree of confidence to reject the notion that there may be no difference. Compared to the single-site data reported by other researchers, the Japanese also do not appear as productive as the most dramatic claims (e.g., the U.S. Department of Commerce study asserting the equivalent of 24,000 SLOC/year for the Japanese). On the other hand, even adjusting for differences in programming language, which other reports have not done, the data presented in Table 13 strongly support the notion that Japanese projects are, at the least, comparable to U.S. counterparts in lines-of-code productivity.

While differences across countries may have implications for long-term competitiveness, of immediate interest to software managers is why differences exist among projects. In particular, can the results be explained by differences in the composition of the systems delivered in Japan versus the U.S.? In order to answer this and other questions, a simple linear model of labor productivity was developed, using variables reflecting differences in the two-country sample that could reasonably be assumed to have an impact on productivity. A country dummy variable (=1 if country = Japan) was also included. The variables chosen are shown in Table 14.

The hypothesized impact of these variables is as follows: Data-processing applications are perceived to be less difficult, and are more likely to use higher level languages than other applications (such as scientific or real-time), and therefore should exhibit higher productivity (Boehm 1981, Jones 1986). Mainframe applications may be less productive than minicomputer or microcomputer applications, as they imply larger, more complex projects than minicomputer applications, and therefore diseconomies of scale may set in (Brooks 1975, Boehm 1981, Banker and Kemerer 1989). In addition, they may add response-time delays due to being a large, shared resource where development may compete with production jobs for machine cycles (Banker et al. 1991). Greater time in the coding phase may be a sign of projects with inadequately specified designs, or so-called "gold-plating" (Boehm 1981), which also would suggest projects with less resulting productivity. Finally, greater code reuse should increase productivity as measured by SLOC (Jones 1984).

The results of the model are shown in Table 15.[7] The model explains about half the variation in productivity for these projects, with signs of the coefficients of the independent variables all in the expected direction. The value for code reuse is significant at the alpha

TABLE 14

Independent Variables in Productivity Regression

Variable	Explanation
x_1	Dummy variable, =1 if country is Japan, else 0
x_2	Dummy variable, =1 if application is data processing, else 0
x_3	Dummy variable, =1 if hardware platform is mainframe, else 0
x_4	Percentage of time in coding phase (expressed in whole numbers)
x_5	Percentage of code reused (expressed in whole numbers)

[6] In order to check the sensitivity of the results to the Fortran conversion, the same test was run on the unadjusted data, which yielded similar results.

[7] The Belsley-Kuh-Welch test of collinearity was run, and no confounding of these results by collinearity is suggested (Belsley et al. 1980). The residuals were plotted against the predicted y values, and the pattern suggested possible heteroscedasticity. However, the results of a Goldfeld-Quandt test on each of the independent variables was that the null hypothesis of homoscedasticity could not be rejected at the alpha = 0.01 level (Pindyck and Rubinfeld 1981, pp. 104–105).

TABLE 15
Productivity Regression Results
(Values of t-statistics shown in parentheses)

SLOC/work-year =	9182 +	$1643x_1$ +	$10450x_2$ −	$7962x_3$ −	$166x_4$ +	$438x_5$
	(1.72)	(0.42)	(2.44)	(−2.24)	(−1.42)	(4.82)

$R^2 = 0.50$.
Adj-$R^2 = 0.42$.
F-stat = 6.730 (0.000).
$n = 40$.

= 0.001 level and the values for data processing application and mainframe platform are significant at the alpha = 0.05 level. The values for coding phase percentage and the country dummy are not significant at the alpha = 0.10 level. The last result is similar to that shown in Table 13, which did not find a significant difference between the productivity means for the Japanese and U.S. samples.

One important point to note about these results is that the U.S. sample had more data-processing systems, which exhibit higher productivity than the more technical systems. Therefore, the nonparametric tests on this sample may understate the productivity differences between the U.S. and Japanese projects, compared with an identically matched sample, to the extent that the U.S. sample has proportionally more data-processing projects.

PROPOSITION 7. *Japanese software projects exhibit fewer defects per lines of code than U.S. projects* (*Table* 1).

Quality is a particularly important performance measure in software since it has long been argued that overall productivity, that is, productivity taking into account the life-cycle maintenance required to fix software defects, is directly related to the quality of the original designs and code (Brooks 1975, Boehm 1981). The quality metric chosen for this research was the number of failures per thousand noncomment source lines of code during the first 12 months of the system's service. Failures were defined as "basic service interruptions or basic service degradations of severity such that correction was not deferrable." Data were available from 20 of the U.S. projects and 11 of the Japanese, and are presented in Table 16.

Similar to the results for productivity, the Japanese projects showed mean and median numbers of failures lower than the U.S. sample (one-half to one-fourth), although these differences were not statistically significant at generally accepted confidence levels (alpha = 0.16). Again, these data do not represent the extremes suggested by some previous research (e.g., the U.S. Department of Commerce study claimed that Japanese error rates were one-tenth the U.S. rates), although the Japanese median does support one prior claim of 0.3 defects/1000 SLOC (Haavind 1986). Overall, while Table 16 indicates a relatively high level of quality in Japanese projects, the variance in the available data is sufficiently high that Proposition 7 cannot be supported with a high degree of confidence.

As with productivity, while verifying the differences across countries provides indications of where "best practice" might be occurring, software managers should be most

TABLE 16
Software Quality
(failures/KSLOC during first 12 months)

	U.S. (median) (n = 20)	Japan (n = 11)	Wilcoxon Rank Sum Z Approximation
Failures/KSLOC	4.44 (.83)	1.96 (.20)	−1.40

TABLE 17
Independent Variables in Quality Regression

Variable	Explanation
x_1	Dummy variable, =1 if country is Japan, else 0
x_2	Size of system in 1000s of Fortran-equivalent SLOC
x_3	Percentage of time in testing phase (expressed in whole numbers)
x_4	Dummy variable, =1 if any testing tool used, else 0

interested in why quality differences exist. Can these results be explained by differences in the composition of the systems delivered in the two countries? In order to answer this question, a simple linear model of quality was developed, using variables reflecting differences in the two-country samples that could reasonably be assumed to have an impact on quality and a dummy variable representing country. The variables chosen are shown in Table 17.

The hypothesized impact of these variables is as follows: The larger a system, the more difficulty in thoroughly testing it. A greater percentage of time in the testing phase should reduce the number of later failures. The only possible effect of mainframes may be the possibly greater availability of testing or debugging tools. However, this can be measured directly, so the hardware platform categorical variable is not included, and in its place is whether any testing or debugging tools were used, which would be expected to improve quality.[8]

The results of the model are shown in Table 18.[9] The interpretation of the model is that more failures/1000 SLOC are present in larger systems (significant at the alpha = 0.001 level). The data support the notion that use of one or more testing tools appears to reduce the error rate (alpha = 0.06), but neither the time spent in the testing phase nor the country dummy were statistically significant. In explaining U.S.-Japanese differences, given that the failure rate is higher in larger systems, and that the size of the Japanese systems in this sample was somewhat larger, the difference between the quality levels in the two countries' projects may again be underestimated in the nonparametric tests, compared to a sample of identically matched systems.

6. Sensitivity Analysis

While the sample size of the data set is large in comparison to other U.S.-Japanese studies, it is still relatively small in statistical terms. In particular, it may appear that the

TABLE 18
Quality Regression Results
(Values of t-statistics shown in parentheses)

$$\underset{}{\text{Failures/KSLOC}} = \underset{}{9.83} - \underset{(-1.33)}{3.06x_1} + \underset{(4.10)}{0.01x_2} - \underset{(-1.38)}{0.13x_3} - \underset{(-1.94)}{4.87x_4}$$
(2.93)

$R^2 = 0.48$.
Adj-$R^2 = 0.40$.
F-stat = 5.897 (0.002).
$n = 31$.

[8] Variables relating to system type (data processing or real-time) were not included in the model, since the relation of system type to quality is not widely agreed upon. Data-processing applications may, in general, have less stringent reliability requirements, and therefore may exhibit lower quality. On the other hand, the perceived greater complexity of real-time systems may make them harder to debug, and therefore their quality may be less. Depending upon which effect dominates, the reliability requirement or the complexity factor, it is unclear what the sign of these variables will be. However, a model including these variables was later run for purposes of sensitivity analysis, and these system-type variables were not found to be significant.

[9] The discussion in footnote 7 applies here as well.

data originate in relatively few firms within each country and that firm-specific effects may be dominant. Actually, this is not believed to be the case simply because even multiple projects from the same firm often come from completely different divisions, which frequently have very different development environments, tailored to individual applications or reflecting variations in management practices. Nonetheless, additional analyses of the statistical tests were performed to evaluate the sensitivity of the results to firm-specific effects.

A. *Nonparametric Tests*

Previous sections presented the results of 17 nonparametric tests. For this section, these tests were rerun, each time excluding one of the six firms that supplied three or more projects. (This turned out to be four of the 11 U.S. firms, and two of the six Japanese firms.) The firms supplying the most data were chosen since, if firm effects existed, it seemed likely to be among this group. This amounted to 102 additional tests, 68 U.S. exclusions and 34 Japanese. In each case, the level of statistical significance was examined to determine whether the exclusion of a single firm's data affected the results.

The results for the U.S. firms are presented in Table 19. As can be seen, of the 12 combinations of tests and firms where the original difference was statistically significant at the alpha ≤ 0.05 level, in eight cases it remained at this level. In four cases, it dropped to the ≤ 0.10 level. These cases were the percentage of time in the coding phase (two cases) and the percentage of full-time testers. In no case did the level of significance fall below 0.10.[10] Regarding previously insignificant differences, in the 56 cases, only two became more significant. Excluding one U.S. firm made the difference in software quality significantly different at the 0.10 level. The interpretation of this is that the excluded U.S. firm's projects were relatively better than the other U.S. firms on average, and excluding them increased the difference between the two countries' samples. In the other case, excluding a different U.S. firm's projects made the difference in productivity significant at the 0.05 level. The interpretation of these results is similar: that one U.S. firm's projects were relatively more productive than the other U.S. firms, and excluding it from the analysis increased the average difference between the two countries' samples.

A similar analysis was done by excluding two of the Japanese firms that supplied three or more projects. These results are presented in Table 20. As can be seen, the impact of excluding a single firm is larger in the Japanese sample, as would be expected given its smaller size and, therefore, the greater contribution of each individual firm's projects to the sample. Of the six test and firm combinations significant at the 0.05 level, two remained at that level, one dropped to the 0.10 level (percentage of time spent in the coding phase), and three dropped below this level (percentage of time spent in the coding phase, and full-time tester percentage [two cases]). In one case, a previously insignificant result

TABLE 19
Sensitivity Analysis of Statistical Significance with Exclusion of Multiple-Project Data from U S Firms

		Original Level	
	alpha	≤ 0.05	>0.05
Level Excluding Firm Multiple-Project Data	≤ 0.05	8	1
	≤ 0.10	4	1
	>0.10	0	54

[10] Actually, the worst case was 0.08.

TABLE 20
Sensitivity Analysis of Statistical Significance with Exclusion of Multiple-Project Data from Japanese Firms

		Original Level	
	alpha	≤0.05	>0.05
Level Excluding Firm Multiple-Project Data	≤0.05	2	0
	≤0.10	1	1
	>0.10	3	27

(programmer experience, original Japanese mean =3.31 years, adjusted Japanese mean =2.58 years, U.S. mean =3.48 years) became significant at alpha = 0.103. However, no changes were observed in either the productivity or quality performance variables.

In summary, the nonparametric test results are relatively robust to the exclusion of any of the six firms who were major data contributors. In just three of 102 cases, previously significant results dropped below the 0.10 level of significance; these were in coding percentage and full-time testing percentage (two cases). In one case, exclusion of a highly productive U.S. firm made the difference between the remaining U.S. firms and the Japanese productivity data significant at the 0.05 level, in favor of the Japanese. On the other hand, the Japanese productivity and quality data do not appear to be strongly influenced by either of the two Japanese firms excluded.

B. *Parametric Tests*

In Section 5, two multivariate linear-regression models were presented, one for productivity and one for quality. Tests of the standard regression assumptions of the absence of either heteroscedasticity or multicollinearity were reported in the previous section. In this section, since there are only two models, the sensitivity of the results to firm-specific effects were tested by sequentially omitting every firm, although any effects are likely to be confined to the cases of firms supplying more than two projects. This analysis for the regression model is summarized in Table 21.

The columns refer to each of the independent variables in the regression: x_1 = country dummy variable, x_2 = data-processing application dummy, x_3 = mainframe dummy, x_4 = coding phase percentage, and x_5 = reused code percentage. The rows correspond to standard levels of statistical significance. The values in the cells are the frequencies of an independent variable possessing the indicated level of statistical significance. The underline indicates the original level with no firm's projects omitted. For example, the country dummy variable was never statistically significant at even the alpha = 0.10 level, no matter which firm's data were omitted. The data processing dummy variable was similarly

TABLE 21
Sensitivity Analysis of Productivity Variable Significance

		Independent Variable				
		x_1	x_2	x_3	x_4	x_5
Significance Level Omitting Each Firm	≤0.05	0	<u>16</u>	<u>12</u>	0	<u>16</u>
	≤0.10	0	1	5	2	0
	>0.10	<u>17</u>	0	0	<u>15</u>	1
Total		17	17	17	17	17

robust, with only one occasion where the omission of a single firm's data (in this case, a U.S. firm) caused the significance level to drop below the original 0.05 level threshold. The mainframe dummy was less robust, with the exclusion of the data from any of five firms (some U.S., some Japanese) causing the level to drop below 0.05 (although never below 0.10). Coding percentage (x_4) showed more movement and, consistent with the nonparametric sensitivity analysis results, its effect seems sensitive to the sample used. Finally, the percentage of code reuse (x_5) is very sensitive to the exclusion of one (Japanese) firm's data. In fact, an examination of the data at the firm level revealed a few projects exhibiting relatively high levels of reuse that contribute disproportionately to this result. Therefore, the effect of code reuse is highly sensitive to the inclusion of one firm's data, and therefore this result should be interpreted cautiously by software development managers and researchers.

A similar analysis was done of the quality model, and the results are shown in Table 22. The columns represent the four independent variables as follows: x_1 = country dummy variable, x_2 = system size, x_3 = testing phase percentage, and x_4 = testing-tool usage dummy. Similar to the productivity results, exclusion of any firm does not make the country dummy significant. Also, the impact of system size remains strongly significant regardless of the removal of any firm from the analysis. Like the coding percentage variable in the productivity model, the testing percentages variable shows a fair amount of movement, and therefore is sensitive to the sample used. Finally, the testing tool dummy variable, originally significant at the alpha = 0.06 level, varies from 0.03 to 0.19 with the removal of data from individual firms. Therefore, this variable is also relatively sensitive to the sample.

In summary, the sensitivity analysis has shown that, while some results in the Japan-U.S. project comparison were influenced by data from single companies, the main findings of this study remain relatively insensitive to the exclusion of data from individual firms. In particular, the main result that Japanese projects are not significantly behind the U.S. in productivity and quality continue to be supported, and, in fact, the exclusion of data from one or two U.S. firms tilts the productivity and quality averages more in favor of the Japanese. No such corollary effect was found by excluding any of the Japanese firms, suggesting relatively uniform performance across the sample of Japanese projects. In terms of the productivity regression model, the data processing dummy result is the most robust. The reuse result is very sensitive to the exclusion of one Japanese firm. In terms of the quality regression model, the size variable is the most robust, being relatively unaffected by the exclusion of any firm. These findings strengthen other indications (such as the greater percentage of data-processing projects in the U.S. sample with associated higher productivity, and larger projects in the Japanese sample with higher defect rates associated positively with larger systems) that the nonparametric tests may reflect a bias in favor of higher productivity and quality for the U.S. projects.

TABLE 22
Sensitivity Analysis of Quality Variable Significance

		Independent Variable			
		x_1	x_2	x_3	x_4
Significance Level Omitting Each Firm	≤0.05	0	14	0	4
	≤0.10	0	0	1	7
	>0.10	14	0	13	3
Total		14	14	14	14

7. Conclusion

This paper began by summarizing an ongoing debate revolving around Japanese performance in software development, an area where U.S. firms have dominated since the beginning of the industry. Anecdotal literature, based on one or two companies or on information from only one site and not analyzed statistically, provided contradictory and ill-supported evidence. Several reports also claimed that the Japanese remain far behind the U.S. in software development, despite progress in several areas. The research presented here provides what appears to be the first review of existing literature as well as the first quantitative data analysis comparing software development practice and performance in the U.S. and Japan.

While a sample of 40 projects can by no means be conclusive, there is enough evidence to provide a preliminary assessment of Japanese project performance. Tables 3 through 18 indicate that: (a) Japanese and U.S. projects develop roughly similar products; (b) Japanese projects work with systems of at least equivalent size; (c) both use nearly identical languages, tools, and hardware platforms; (d) both use personnel with comparable years of experience; and (e) at least equivalent levels of reuse, productivity, and quality in the Japanese projects suggest similar and possibly lower development costs over time for individual systems. As a result, this study finds no support for the belief that Japanese skills in software development still appear to be inferior overall to those in the U.S.

Several specific and potentially important differences did surface in the analyses. While individual years of work experience appeared to be similar between the Japanese and the U.S. samples, projects varied significantly in how they used personnel. The Japanese spent less time than their U.S. counterparts on coding, a relatively routine part of software development, and more time on design. Examination of the data further suggested that code reuse was associated with higher productivity. The study also revealed that larger projects tended to have more defects and the Japanese projects had fewer defects, in spite of having slightly larger systems in this sample. Use of testing tools was associated with lower error rates.

While the project data indicated at least a parity in managing the process of software development, it still may not be easy for Japanese companies to emerge as strong rivals of U.S. (or European) firms outside Japan. Local service and business relationships, as well as fluency in languages and practices, may be as important as expertise in software engineering. Firms also need a surplus of skilled personnel versed in foreign languages and practices to develop custom applications or packages for export. The Japanese did not appear to have this surplus in the 1980s, although companies were beginning to compete abroad in programming contracts, such as for telecommunications systems, as well as to service overseas subsidiaries of Japanese companies and provide a few special products for foreign markets, such as operating systems to accompany exported hardware.

In addition, case studies demonstrate that the leading Japanese computer producers are paying increasing attention to software functionality and ease of use, as well as reusability, automation, and other measures to improve productivity and quality simultaneously (Cusumano 1991). Judging from the high performance standards this study indicates Japanese producers have already set for themselves, and their tendency to pay more attention to design and testing rather than routine coding operations, it is likely the Japanese will continue to improve their capabilities and potential for international competition in software.

In conclusion, this comparison of Japanese and U.S. projects supports a view of software development that, similar to conventional engineering or manufacturing disciplines, associates reuse of components, tool support and automation of routine tasks, as well as rigorous inspection and testing, with high levels of productivity and quality in final products. In other words, software does not appear to be a technology inherently restricted

to a loosely organized art or craft approach to development. Disciplined engineering, production, and quality-management practices may well improve project performance in software. It follows that there appears to be no empirical or theoretical basis to argue that Japanese firms should encounter unique obstacles to performing well with this technology, although larger-sample studies, longitudinal analyses to identify specific areas and rates of improvement, as well as more detailed probing of practices and products, are needed to better understand the many factors that affect project performance in software development.[11]

[11] The authors would like to thank each company and individual who participated in this study, since their cooperation made this effort possible. The authors also gratefully acknowledge the research assistance of Kent Wallgren, who assisted in the data collection and preliminary analysis as part of a Masters' Thesis Project at the M.I.T. Sloan School of Management in 1987–1988. Helpful comments were received on earlier drafts from W. Orlikowski, N. Venkatraman, D. Zweig, and three anonymous referees.

Appendix A. Companies and Product Areas Participating in the Study

U.S. Sites/Product Areas
Amdahl/Product Software
Amdahl/Engineering Software
AT&T Bell Laboratories/Switching
AT&T Bell Laboratories/Communications Database
AT&T Bell Laboratories/Transaction Processing
Computervision/Computer-Aided Manufacturing
Computervision/Drafting
Computervision/Research & Development
Financial Planning Technologies/Planning Systems
Harris Corporation/Government Support Systems (3 Projects)
Hewlett-Packard/Medical Division (2 Projects)
Yokogawa/Hewlett-Packard/Medical Products
Honeywell/Corporate Systems (3 Projects)
Hughes Aircraft/Communications & Data Processing (3 Projects)
International Business Machines/Basic Systems Software
International Business Machines/Systems Integration Division
Unisys/Computer Systems (3 Projects)
Bell Communications Research/Applications
Bell Communications Research/Software Technology & Systems
Japanese Sites/Product Areas
Fujitsu/Communications Software
Fujitsu/Basic Software (2 Projects)
Fujitsu/Applications Software
Hitachi/Basic Software
Hitachi/Applications Software
Hitachi/Switching Software
Hitachi Software Engineering/Financial Systems
Hitachi Software Engineering/Operating Systems
Kozo Keikaku/Computer-Aided Design
Mitsubishi Electric/Communications Software
Mitsubishi Electric/Systems Software
Mitsubishi Electric/Power & Industrial Systems Software
Nippon Business Consultant/System Software
Nippon Electronic Development/Communications Systems
Nippon Electronics Development/Information Service Systems
Nippon Systemware/System Software
Nippon Telegraph & Telephone/System Software
Nippon Telegraph & Telephone/Network Systems
Nippon Telegraph & Telephone/Applications

Appendix B. Sample Description

	Development Sites Receiving Forms	Number Returned	Number Discarded	Projects Analyzed
U.S.	44*	28*	4*	24
Japan	26	20	4	16
Total	70	48	8	40

* Includes one U.S.-Japanese joint venture submitted by a U.S. firm.

References

ALBRECHT, A. J. AND JOHN GAFFNEY, JR., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, SE-9 (November 1983), 639–648.

ABEGGLEN, JAMES C. AND GEORGE STALK, JR., *Kaisha: The Japanese Corporation*, Basic Books, New York, 1985.

ARDEN, BRUCE, *What Can Be Automated?*, MIT Press, Cambridge, MA, 1980.

BANKER, R., S. DATAR AND C. KEMERER, "Factors Affecting Software Maintenance Productivity: An Exploratory Study," *Proc. of the 8th International Conference on Information Systems*, Pittsburgh, PA, December, 1987, 160–175.

———, "A Model to Evaluate Variables Impacting Productivity on Software Maintenance Projects," in press, *Management Science*, 37 (January 1991).

BANKER, R., AND C. KEMERER, "Scale Economies in New Software Development," *IEEE Transactions on Software Engineering*, 15 (October 1989) 1199–1205.

BELSLEY, D., E. KUH AND R. WELSCH, *Regression Diagnostics*, John Wiley and Sons, New York, 1980.

BELADY, LASZLO A., "The Japanese and Software: Is It a Good Match?" *IEEE Computer* (June 1986), 57–61.

BOEHM, BARRY W., *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, N.J., 1981.

———, "Improving Software Productivity," *IEEE Computer* (September 1987), 43–57.

BRADLEY, JAMES V., *Distribution-Free Statistical Tests*, Prentice-Hall, Englewood Cliffs, NJ, 1968.

BROOKS, FREDERICK P., *The Mythical Man-Month*, Addison-Wesley, Reading, MA, 1975.

———, "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Computer* (April 1987), 10–19.

CHRYSLER, E., "Some Basic Determinants of Computer Programming Productivity," *Communications of the ACM*, 21 (June 1978), 472–483.

COLE, ROBERT E., *Work, Mobility, and Participation: A Comparative Study of American and Japanese Industry*, University of California Press, Berkeley and Los Angeles, 1979.

CONTE, S., H. DUNSMORE AND V. SHEN, *Software Engineering Metrics and Models*, Benjamin/Cummings, Reading, MA, 1986.

CUSUMANO, MICHAEL A., *The Japanese Automobile Industry*, Harvard University Press, Cambridge, MA, 1985.

———, "Hardware and Software Customer Satisfaction in Japan: A Comparison of U.S. and Japanese Vendors," *MIT Sloan School of Management*, Working Paper #2101-88, December 1988. (See Cusumano 1991).

———, "The Software Factory: A Historical Interpretation," *IEEE Software* (March 1989), 23–30.

———, *Japan's Software Factories*, Oxford University Press, New York and London, 1991.

Datamation, "The Datamation 100" and "Company Profiles" (1 June 1985), 58–120.

FRANK, WERNER L., *Critical Issues in Software*, John Wiley and Sons, New York, 1983.

FREEMAN, P., ED., *Tutorial: Software Reusability*, IEEE Computer Society Press, Washington, D.C., 1987.

GAFFNEY, JOHN E., JR., "A Microanalysis Methodology for Assessment of Software Development Costs," in R. Goldberg and H. Lorin (Eds.), *The Economics of Information Processing*, John Wiley & Sons, New York, 1982, 2, 177–185.

GAMAOTA, GEORGE AND WENDY FRIEMAN, *Gaining Ground: Japan's Strides in Science and Technology*, Ballinger, Cambridge, MA, 1988.

HAAVIND, ROBERT, "Tools for Compatibility," *High Technology* (August 1986), 34–42.

HAUPTMAN, OSCAR, "Influence of Task Type on the Relationship Between Communication and Performance: The Case of Software Development," *R&D Management*, 16 (April 1986), 127–139.

HENDERSON, JOHN AND JAY COOPRIDER, "Dimensions of I/S Planning and Design Technology," M.I.T. Center for Information Systems Research, Working Paper #181, September 1988.

HUNKE, H., ED., *Software Engineering Environments*, North-Holland, Amsterdam, 1981.

JOHNSON, COLIN, *Electronic Engineering Times*, "Software in Japan" (11 February 1985), 1.

JOHO SABISU SANGYO KYOKAI (JAPAN INFORMATION SERVICE INDUSTRY ASSOCIATION), *Joho Sabisu Sangyo hakusho* 1987 (*White Paper of Information Service Industry* 1987), Tokyo, Joho Sabisu Sangyo Kyokai, 1987.

JONES, T. CAPERS, "Reusability in Programming: A Survey of the State of the Art," *IEEE Transactions on Software Engineering*, SE-10 (September 1984), 488–494.

———, *Programming Productivity*, McGraw-Hill, New York, 1986.

———, "A New Look at Languages," *Computerworld* (7 November 1988), 97–103.

KEMERER, CHRIS F., "An Empirical Validation of Software Cost Estimation Models," *Communications of the ACM*, 30 (May 1987), 416–425.

———, "Software Production Economics: Theoretical Models and Practical Tools," *ACM 27th Technical Symposium Proceedings*, Gaithersburg, Maryland, June 1988.

KIM, K. H., "A Look at Japan's Development of Software Engineering Technology," *IEEE Computer* (May 1983), 26–37.

KIRIU, HIROSHI, *Sofutouea sangyo no jitsuzo* (*The actual state of the software industry*), Tokyo, Nikkan Shobo, 1986.

KISHIDA, KOUICHI, et al., "Quality-Assurance Technology in Japan," *IEEE Software* (September 1987), 11–18.

KRIEBEL, C. H. AND A. RAVIV, "An Economics Approach to Modeling the Productivity of Computer Systems," *Management Sci.*, 26 (1980), 297–311.

LECHT, CHARLES P., "Japanese Software No Threat," *Computerworld* (8 May 1989), 21.

LEWIS, GEOFF, "Computers: Japan Comes on Strong," *Business Week* (23 October 1989), 104–112.

MATSUMOTO, YOSHIRO, "A Software Factory: An Overall Approach to Software Production," in Peter Freeman, Ed., *Tutorial: Software Reusability*, Institute of Electrical and Electronics Engineers, Washington, D.C., 1987, 155–178.

MCKEEN, J. D., "Successful Development Strategies for Business Application Systems," *MIS Quart.*, 7 (September 1983), 47–65.

NAUR, PETER AND BRIAN RANDELL, EDS., *Software Engineering: Report on a Conference Sponsored by the NATO Science Committee*, Scientific Affairs Division, NATO, Brussels, January 1969.

Office of Technology Assessment, U.S. Congress, *International Competition in Services*, U.S. Government Printing Office, Washington, D.C., July 1987.

PINDYCK, R. S. AND RUBINFELD, D. L., *Econometric Models and Economic Forecasts*, McGraw-Hill Book Company, New York, 1981.

PUTNAM, L. H., "General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transactions on Software Engineering* 4 (July 1978), 345–361.

RAMAMOORTHY, C. V., et al., "Software Engineering: Problems and Perspectives," *IEEE Computer* (October 1984), 191–209.

RIFKIN, GLENN AND J. A. SAVAGE, "Is U.S. Ready for Japan Software Push," *Computerworld* (8 May 1989), 1.

SAKAI, TOSHIO, "Software: The New Driving Force," *Business Week* (27 February 1984), 96–97.

SCHONBERGER, RICHARD J., *Japanese Manufacturing Techniques*, The Free Press, New York, 1982.

SELBY, RICHARD, "Quantitative Studies of Software Reuse," in Ted J. Biggerstaff and Alan J. Perlis (Eds.) *Software Reusability*, ACM Press, New York, 2, 1989.

SHOOMAN, MARTIN, *Software Engineering: Design, Reliability, and Management*, McGraw-Hill, New York, 1983.

STABELL, C. B., "Office Productivity: A Macroeconomic Framework for Empirical Research," *Office Technology and People*, 1 (March 1982), 91–106.

STANDISH, THOMAS A., "An Essay on Software Reuse," *IEEE Transactions on Software Engineering*, SE-10 (September 1984), 494–497.

TRACZ, WILL, "Software Reuse Myths," *Software Engineering Notes*, 13 (January 1988), 17–21.

TAJIMA, DENJI AND TOMOO MATSUBARA, "The Computer Software Industry in Japan," *IEEE Computer* (May 1981), 89–96.

U.S. DEPARTMENT OF COMMERCE, *A Competitive Assessment of the U.S. Software Industry*, International Trade Administration, U.S. Department of Commerce, Washington, D.C.,1984.

U.S. DEPARTMENT OF LABOR, BUREAU OF LABOR STATISTICS, *Monthly Labor Review*, Washington, D.C., 1989.

UTTAL, BRO, "Japan's Persistent Software Gap," *Fortune*, 15 October 1984), 151–160.

VOGEL, EZRA F., *Japan as Number 1: Lessons for America*, Harvard University Press, Cambridge, MA, 1979.

WALSTON, C. E. AND C. P. FELIX, "A Method of Programming Measurement and Estimation," *IBM Systems J.*, 16 (January 1977), 54–73.

WOODFIELD, S., D. EMBLEY AND D. SCOTT, "Can Programmers Reuse Software?" *IEEE Software* (July 1987), 52–59.

ZELKOWITZ, MARVIN V. et al., "Software Engineering Practices in the U.S. and Japan," *IEEE Computer* (June 1984), 57–66.