A QUANTITATIVE ANALYSIS OF U.S. AND JAPANESE SOFTWARE-ENGINEERING PRACTICE AND PERFORMANCE

Michael A. Cusumano and Chris F. Kemerer

Sloan School of Management
Massachusetts Institute of Technology

Sloan Working Paper # 3022-89 BPS/MS

July 1989

For Presentation at TIMS XXIX, July 23-26, 1989
Osaka, Japan

Abstract

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 divided between assertions of Japanese or U.S. superiority 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. firms 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 programmers 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.

[SOFTWARE DEVELOPMENT PRODUCTIVITY; SOFTWARE QUALITY; SOFTWARE REUSE; JAPANESE SOFTWARE; JAPANESE MANAGEMENT METHODS]

I. Introduction[1]

Japanese firms have become well-known for their high levels of productivity and reliability in manufacturing, their quality in product engineering, and the increasing sophistication and diversity of their products. They have competed effectively in a broad range of industries, with products of varying degrees of complexity in design, engineering, and assembly: steel, ships, automobiles, televisions, video recorders, machine tools, semiconductors, and computer hardware, to name a few. Yet, despite this impressive record of successes, some researchers doubt whether the Japanese will duplicate their achievements with new, still-evolving technologies, where customers and producers have yet to define product or process standards. A major challenge -- and the subject of this comparative paper -- is the design and production of computer software.

Managing software development has been problematic since the beginning of the industry in the late 1950s, when programmable computers first appeared. Rapidly changing and rising demand has consistently exceeded the ability of educational institutions and companies to train enough skilled programmers to meet market needs. Attempts to manage the development process more effectively have seemed woefully inadequate compared to dramatic improvements in hardware. Many software producers continue to experience cost and schedule overruns as the rule rather than the exception, while huge variations in customer requirements and programmer productivity have further complicated managerial tasks. As a result, characteristics of the industry and the

---

[1]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 an early draft from W. Orlikowski, N. Venkatraman, and D. Zweig.

technology produced a situation referred to as the "software crisis" as long ago as 1969 (Hunke 1981, Frank 1983). Most of the problems cited in the 1960s continued to plague software producers in the 1980s (Arden 1980, Boehm 1987, Ramamoorthy 1984, Brooks 1987).

The primary objectives of this paper were to compile comparisons of Japanese and U.S. project performance in software and then offer some exploratory but quantitative data in order to shed some light on a debate that, up to the present, has proceeded largely in an anecdotal and qualitative fashion. The disagreements center around the efficacy of the Japanese in software development, specifically, whether Japanese firms have been able to transfer their skills in engineering, production, and organization management to this relatively new technology.

This paper is structured as follows: The second section reviews existing literature in an attempt to identify what statements observers have made regarding the Japanese and U.S. in software development. The third section explains the research methodology followed to confirm or deny several propositions indicated in the literature. This methodology consisted of the collection 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. The concluding section summarizes the results and limitations of the study, as well as implications for future research.

II. 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 claims that the Japanese were significantly behind the U.S. overall in

software development, especially in products. The arguments consisted of general statements that Japanese firms relied on tools and techniques adopted from the U.S. and Europe, and they suffered from a severe shortage of skilled programmers. Japanese programmers also seemed to lack creativity and the ability to invent new products or offer sophisticated products, especially software packages. Japanese software also seemed more costly than U.S. programs. In addition, the Japanese were behind in basic research. To many authors, these reasons made it unlikely Japanese firms would compete effectively in the U.S. software market (Table 1).

On the other side of the debate were arguments that Japanese firms were ahead in areas related to the **process** of software development: productivity, quality (defect) levels, tool usage, and reusability, as well as discipline, teamwork, planning, and management. Some authors also believed 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 (see, for example, Belady 1986).[2] In the **product** area, there were more disagreements, although some observers felt the Japanese were at least equal to the U.S. in custom applications programming 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 2).

Comments from specific articles reflects the qualitative and sometimes contradictory nature of the debate. For example, a 1983 publication based on a

---

[2] Although the literature on Japanese management styles and employee behavior is too long to cite comprehensively, examples include Vogel 1979, Cole 1979, Schonberger 1982, and Abegglen and Stalk 1985.

visit to 10 Japanese software producers and R&D organizations, sponsored in part by the U.S. Office of Naval Research (Kim 1983), claimed Toshiba's Software Factory was "one of the most advanced real-time software engineering organizations in the world," with productivity averaging 2870 instructions per programmer month in 1981, an extremely high level compared to U.S. figures. The author thought that the productivity advantage stemmed from a Japanese lead in developing and using integrated tool sets. In the product area, the Japanese appeared behind in package development but advanced in real-time software and artificial intelligence applications (primarily for language translation).

A 1986 article provided more numbers from Toshiba: 65% reuse in delivered code and productivity of 2000 lines per month per programmer, with merely 0.3 defects per 1000 lines of code (Haavind 1986). Once again, however, the author did not describe the number and nature of the projects covered, or what specifically the numbers measured. These were important qualifications: For example, a 1987 article by the Toshiba manager who headed the software factory noted all productivity figures were annual averages converted to "equivalent assembler source lines of code," not actual number of lines written. There were also various defect levels, depending on customer requirements and length of testing (Matsumoto 1987). But while the numbers were difficult to assign precisely, other articles continued to report higher levels of reuse in Japan (Standish 1984, Tracz 1988) as well as greater management emphasis on designing for reuse and reusing code when developing new programs (Cusumano 1989).

A 1984 article, based on a questionnaire survey and site visits, examined 30 software production facilities at 13 Japanese and 13 U.S. companies (Zelkowitz et al. 1984). The authors found similar technology for software

development in the two countries, with Japanese firms relying on tools and techniques from the U.S. or Europe. However, the Japanese seemed to use tools more widely, perhaps because tool development came out of company overhead rather than project budgets. This study also found the Japanese conducted extensive analyses of the causes of software defects, although it left the question open whether Japanese firms were systematically better in quality, productivity, or other areas of software development.

In 1984, the U.S. Department of Commerce published an analysis of U.S. competitiveness in software that cited both weaknesses and strengths in the Japanese industry (U.S. Department of Commerce 1984). This report maintained that the Japanese were "far behind the U.S. in basic research and advanced development of software," and "have never developed a programming language or an operating system that has become a **de facto** standard internationally." Nonetheless, the authors recognized the Japanese were making rapid progress in tool usage and product engineering, especially along the "software factory" model, and were overtaking the U.S. in their efforts to move software production beyond the "craft" stage. This report also contained some quantitative data: "Japanese programmers average 2000 lines of code per month (versus less than 300 lines per month for U.S. programmers) and have one-tenth the error rate (defects) of their U.S. counterparts" (p. 11). However, the Department of Commerce offered no information to allow the reader to evaluate and compare the numbers it cited. The report did not, for example, reveal what kind and how many projects, what phases of the development cycle, or what levels of reused code these "averages" reflected.

Other articles continued to refer to high levels of productivity, reuse, and quality in Japanese software, but general backwardness in products and research. A 1984 article noted the rise of "software factories" in Japan and

their highly systematic approach to software production, as well as Japanese excellence in graphics and embedded software, such as in industrial and consumer products. But it concluded that Japanese software "is less sophisticated and more costly to produce than Western software" (Uttal 1984). Another 1984 article noted Japanese strengths in certain types of software products -- video games, super-computer programs, large-scale banking and airline-reservation systems -- as well as a possible 10% to 15% edge in productivity. However, the authors concluded the Japanese industry was too small (one-quarter of U.S. sales) to compete effectively with the U.S. (Sakai 1984).

A 1987 study by the U.S. Office of Technology Assessment (OTA) on U.S. competitiveness in service industries (Office of Technology Assessment 1987) agreed the Japanese, in the long run, would "emerge as the primary U.S. competitor in software." OTA concluded, however, that "Japan remains substantially behind in software, with poor applications packages -- along with limited sales and service networks...." Nonetheless, Japan seemed to have emerging strengths: The researchers asserted that Japanese systems software, based on U.S. products, "is usually considered to be quite good," and Japanese language processing software was sophisticated. Furthermore, while custom programming was far less efficient than package use and still constituted the bulk of the Japanese software market, Japanese firms such as Toshiba seemed to excel at producing complex software efficiently and with many fewer defects than in the U.S. (pp. 160-166).

While information on products was scarce and difficult to evaluate, a 1988 survey of 20,000 Japanese computer users by a Japanese journal indicated that Japanese firms were to some extent superior to U.S. firms competing in Japan in custom applications and software maintenance, in addition to offering lower

prices for leased software and system-engineering services. But Japanese customers were less satisfied with Japanese products, compared to U.S. products, in basic systems software and office systems (Nikkei Computer March 1988, September 1988; Cusumano 1988).

A study supported by the U.S. National Science Foundation to evaluate Japanese technology (JTECH Project) concluded as well that, while Japanese firms seemed excellent in product engineering, reusability, tool use, and quality, Japan overall was behind in basic research and advanced development for software (Gamaota and Frieman 1988). A 1989 article made broader claims, noting that the Japanese software industry was still "small and not very visible," and perhaps two decades behind the U.S. (Rifkin and Savage 1989). Yet another 1989 essay maintained that, while the Japanese were strong in applications systems for their home market, they appeared weak in other product areas. This author argued that, in general, the Japanese suffered from a severe shortage of skilled programmers and had few software houses that were financially strong enough to compete with U.S. firms in the American market (Lecht 1989).

Perhaps the most glaring defect in this literature has been its anecdotal nature -- general comments that are difficult to test, such as assertions that Japanese products lacked sophistication or creativity; and assertions that relied on information from one or a few firms, with little or no attempt to collect quantitative data systematically. Part of the reason for a debate of this type 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 or articles cited provided several suggestions of differences between Japanese and U.S. practice and performance in software development:

Proposition 1: *Japanese software developers copy or rely on Western tools and techniques (Table 1).*

Proposition 2: *Japanese software is less sophisticated than U.S. software (Table 1).*

Proposition 3: *Japanese software development managers are ahead in project management (Table 2).*

Proposition 4: *Japanese software developers make greater use of reused code (Table 2).*

Proposition 5: *Japanese software developers are ahead in tool usage (Table 2).*

Proposition 6: *Japanese software developers are ahead in software productivity (Table 2).*

Proposition 7: *Japanese software developers are ahead in software quality (Table 2).*

Analysis of commonly-defined data collected as part of this research should either support or question these propositions, as well as indicate if there are significant interrelationships among some of the variables.

III. Research Methodology

The methodology adopted to compare Japanese and U.S. firms was to collect data on standardized forms from individual development sites for particular software systems. The first task was to compile a list of major software producers and development sites in each country that also were comparable in terms of product 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 was to identify managers of software development at these sites willing to complete a standardized data-collection form on one or more systems 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 4 of the U.S. surveys and 4 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 or 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.[3]

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

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

U.S. firms in particular, many 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 firms, where data collection appeared to be more routine and thorough. Also, some firms in both countries, even after initially agreeing to submit data, decided 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 systems 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. Nevertheless, it is believed to 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 software-development management. It also provides a unique opportunity to assess quantitatively the claims and counter claims raised by anecdotal reports over the last decade.

IV. Descriptive Data

In this section the Japanese and U.S. samples are compared in terms of a number of descriptive dimensions: application type, programming language, hardware platform, and size.

The data-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 will follow include variables that represent possible differences in application mix across the two samples.

Proposition 1: Japanese software developers copy or rely on Western tools and techniques (Table 1).

Lists of applications, programming languages, and hardware platforms, as well as the tools used in software development, indicate that Japanese firms have largely followed the lead of the U.S. and Europe, where this technology was invented. For example, primary application languages are quite consistent across the two countries (see Table 4a). The U.S. firms 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 hardware platform on which they were delivered are virtually identical, even though the Japanese firms have a slightly greater percentage of microcomputer implementations, and the U.S. firms a small advantage in mainframes (Table 5). More detailed comparisons of tools used discussed below (sees Tables 11 and 12) also lend support to Proposition 1 and to the more specific statement that, in terms of applications, languages, and hardware platforms, as well as tools, there do not appear to be significant differences between the U.S. and Japanese samples under analysis in this paper.

Proposition 2: Japanese software is less sophisticated than U.S. software (Table 1).

The term "sophistication" is vague 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 known to these authors has directly compared Japanese 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 is a strong association between the complexity of software and the size of a system, then the data collected make it possible to compare the Japanese and U.S. systems in an area related to sophistication.

The conventional measure for size in a software project is the number of non-comment source lines of code, or SLOC that are 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 by this statement from a recent article by 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 non-comment SLOC as the output size metric and

work-years as the input size metric.[4] 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 the nonparametric Wilcoxon rank sum test (equivalent to a Mann-Whitney U test) were that no statistically significant difference was found (Bradley 1968: 105-114).

One possible concern with the use of SLOC metrics for inter-project 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: 49, Jones 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 language 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

---

[4]One possible concern about such a measure is the reputation of Japanese firms for long work days. (e.g., 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 system 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: 118-119, 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).

Japanese software is less sophisticated than U.S., at least to the degree that system size reflects sophistication.

V. Process Data

In addition to the descriptive data described in Section IV, data were collected on the process of software development as practiced in the U.S. and Japan. These data consist of both the labor (project staffing) and capital (tool usage) inputs to the software development process.

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 development managers are ahead in project management (Table 2).

The literature suggested the Japanese were strong in various areas of project management, such as planning, discipline, and teamwork. The data collected focused more on performance measures, such as productivity and quality (defects), which probably reflect management skills, but did not directly define areas of management and attempt measurements. However, the form did request data on how projects used personnel, such as the allocation of effort

across the systems development life-cycle, and this provides some insight into project management. In particular, 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).

Average data relating to life-cycle emphasis as collected on the individual systems 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 = .01 level) time in the early life cycle phase, and significantly less (at the alpha = .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. The data from Table 8 suggest hypotheses relating to performance measures that will be tested in Section VI below -- namely, that the Japanese firms in this sample spend a greater percentage of time in the design and test phases (although the latter difference was not statistically significant).

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 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. firms. This is another difference in project management that supports suggestions by other researchers that Japanese firms excel in defect analysis (Zelkowitz 1984) and other aspects of quality control (see Table 2).

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 developers make greater use of reused code (Table 2).

The issue of code reuse has received a tremendous amount of attention recently as a potential "silver bullet" for the software crisis (Brooks 1987).[5] Unfortunately, with only a few exceptions (e.g. Selby 1988), little empirical data has been published on actual reuse in industrial settings. The data for the U.S. and Japanese firms presented in this paper are shown in Table 10. (Since industry experts have theorized that high degrees of software reuse should greatly improve nominal productivity levels (Jones 1986), but without empirical data to demonstrate this, Appendix C presents the results of a test of this relationship using data from the U.S. and Japanese systems.)

Consistent with previous reports (see Table 2), the Japanese firms seem to exhibit higher levels of reuse than U.S. firms. 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

---

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

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 motion that there may be no difference in reuse levels in this sample. 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 developers are ahead in tool usage (Table 2).

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:

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 firms in each of the two countries.[6] This gross level of analysis suggests that Japanese and U.S. firms 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 numbers of firms using any given type of tool become very small, and therefore standard statistical tests seem inappropriate. Examination of the table indicates that 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 firms report using tools to support these activities, although the Japanese report greater use of automatic flowcharting tools. In coding, Japanese firms 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. firms reported using either code generators or reusable code libraries (tools developed in the U.S. and promoted by U.S. experts in software engineering -- see, for example, Boehm 1981, Brooks 1987).

Note that zero use of code libraries is not necessarily inconsistent with the code reuse numbers cited earlier. Several studies describe 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

---

[6]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.

how reuse is supported institutionally has been suggested as a factor that has an impact on these costs and benefits (Cusumano 1989, Matsumoto 1987).

While data on testing or debugging tools is similar across the U.S. and Japanese firms, the percentage of respondents claiming use of these tools was relatively high (roughly two-thirds), compared to data cited by Zelkowitz et al., which shows only 27% claiming use (Zelkowitz, et al., 1984). The difference in these two observations may stem from the five years difference 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 strong support to Proposition 5. The Japanese projects did claim far more extensive use of coding utilities and used automated flowcharting 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.

VI. 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 in the introduction to this paper, the literature is divided between those who feel the Japanese are behind and unlikely to be a threat in software (Table 1) and those who feel the Japanese are already superior in some respects to the U.S. (Table 2). Thus far, authors have supported claims with descriptions or numbers from one or two Japanese sites, which may or may not be exceptional. This section presents the results of the quantitative analysis of productivity and quality on the current sample.

Proposition 6: Japanese software developers are ahead in software productivity (Table 2).

Productivity is defined here as non-comment Fortran equivalent source lines of code per work year (Jones 1986, 1988). Table 13 presents the comparison of the U.S. and Japanese firms along this dimension.[7] Both the mean and median for the Japanese firms are higher (58% to 71%) than the U.S. numbers, although the differences are not statistically significant at usual levels (alpha = .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 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), but they appear better than some modest estimates (e.g., the 1984 claim by Sakai of only a 10% to 15% advantage for Japan). While other studies' numbers did not take programming-language differences into account, after making these adjustments, the data presented in Table 13 strongly support the notion that, at the least, Japanese systems developers are by no means behind their U.S. counterparts in terms of 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 and between the samples from Japan and the U.S. In particular, can Japanese results be explained by differences in the composition

---

[7] 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.

of the systems delivered in the two countries? 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. 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. 1989).

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 will increase productivity as measured by SLOC (Jones 1984). The results of the model are shown in Table 15.

This model explains about half of the variation in productivity in these data. The signs of the coefficients of the independent variables are all in the expected direction. The value for code reuse is significant at the alpha<.001 level, the values for data processing application and mainframe platform are significant at the alpha=.05 level, and the value for coding phase percentage is significant at the alpha=.10 level.

In terms of explaining U.S./Japanese differences, the greater code reuse

and less time spent during coding clearly seems to aid the Japanese firms. The U.S. had more data-processing systems, which exhibit higher productivity than the more technical systems. Therefore, this sample may tend to understate the U.S./Japan productivity differences (due to the U.S. sample having proportionally more data-processing projects) compared with an identically matched sample.[8]

Proposition 7: Japanese software developers are ahead in software quality (Table 2).

Quality is a particularly important performance measure 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 non-comment 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. firms and 11 of the Japanese firms, and are presented in Table 16.

Similar to the results for productivity, the Japanese firms showed mean and median numbers of failures lower than the U.S. firms (one-half to one-fourth), although these differences were not statistically significant at generally accepted confidence levels (alpha = .16). Again, these data do not represent

---

[8]The Belsley-Kuh-Welch test of collinearity was run, and no confounding of these results by collinearity is suggested (Belsley 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=.01 level (Pindyck and Rubinfeld 1981: 104-105).

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). However, the Japanese median does support a prior claim of .3 defects/1000 SLOC (Haavind 1986). Therefore, the data suggest support for Proposition 7, but cannot be used with a high degree of confidence to reject the notion that there may be no difference.

As for productivity, while verifying the differences across countries provides indications of where "best practice" might be occurring, software managers should be most 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 sample that could reasonably be assumed to have an impact on quality. 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/debugging tools. However, this can be measured directly, so the hardware platform categorical variable is not included, and in its place is the number of testing/debugging tools used, which would be expected to improve quality.[9]

---

[9]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 run for purposes of sensitivity analysis, and these system-

The results of the model are shown in Table 18. The interpretation of the model is that more failures/1000 SLOC are present in larger systems (significant at the alpha=.001 level). A greater percentage of time spent in the testing phase reduces the error rate, as does a greater use of testing tools (both results are significant at approximately the alpha=.10 level).[10] In terms of explaining U.S./Japanese differences, given that the failure rate is higher in larger systems, and given the somewhat larger size of the Japanese systems in this sample, the difference between the two countries may tend to be underestimated compared to a sample of identically matched systems.

VII. Conclusions

This paper began by summarizing an ongoing debate revolving around how well the Japanese are performing 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 confusing evidence. Assertions ranged from claims that Japanese firms were already vastly superior to U.S. firms in software productivity and quality to suggestions the Japanese were still far behind the U.S. in this industry by a variety of measures. 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.

Data from 40 systems do not support propositions that there are major differences between U.S. and Japanese performance in software development, either in a positive or negative sense. Japanese and U.S. firms appeared to use

---

type variables were not found to be significant.

[10]The discussion in footnote 8 applies here as well.

similar tools, develop systems of comparable sophistication, and displayed statistically comparable levels of reuse, productivity, and quality. However, to those who argued that Japan lagged far behind the U.S., it seems clear that Japanese software developers are, at a minimum, as good, in productivity, quality, and management, as their U.S. counterparts. In fact, it would be a mistake for U.S. firms to be complacent in this industry. Japanese performance, as reflected in the collected metrics, on average appears to be superior to the U.S., although differences were not significant statistically, reflecting high variances as well as the small size of the final sample. These are, unfortunately, common problems in the collection of empirical data on software engineering (Jones 1986). Nonetheless, the study revealed potentially emerging and important differences.

In project management, individual years of experience were similar, in contrast to claims that the Japanese lacked experienced staff, compared to the U.S. Yet there were significant differences in how these personnel were used. The Japanese spent less time than their U.S. counterparts on coding, a relatively routine part of software development, and more time on design. Productivity, measured as non-comment lines of code per work year and adjusted for different languages, was about 60% to 70% higher in the Japanese projects, in spite of some application differences in the sample that may have favored the U.S. Analysis of the data further suggested that code reuse and less time spent in coding boosted Japanese productivity averages. The number of major defects per 1000 lines of code in the first twelve months after delivery in Japanese software products was half or less than in the U.S. sample. This again suggests a difference of potential importance for managers interested both in good practice and potential competition from Japan. The analysis also revealed that (1) larger projects tended to have more defects, and (2) the

Japanese tended to have fewer defects in spite of having slightly larger systems on average in this sample. More time spent in testing, and greater use of testing tools, was shown to be associated with lower error rates.

Larger-sample studies, longitudinal analyses to identify rates of improvement, as well as more exploration of differences in management and development practices, are needed to explore further the comparison of Japanese and U.S. software producers. International comparisons are important both to probe the reasons behind good practice, wherever this may occur, and to evaluate the performance of individual firms and projects.

As for future competition from the Japanese in software, detailed studies of companies and facilities reveal that the Japanese are paying increasing attention to product functionality and ease of use (Cusumano 1989), as well as reusability and automation. Combined with greater attention already being paid to design and testing, and their apparent strengths in managing the process of software development, it is likely the Japanese will continue to improve their capabilities and potential for international competition in software. It is an open question to what extent companies from Japan will emerge as strong rivals of U.S. or European firms outside Japan, where local service and relationships, as well as fluency in local languages and business practices, may be as important as expertise in systems development. Japanese firms also need to have a surplus of skilled software personnel versed in foreign languages and practices to develop custom applications or packages for export, and they did not appear to have this surplus in the 1980s.

In conclusion, this study suggests that Japanese and U.S. performance in software development at the moment is more alike than different, and projects from both countries display considerable variability. However, the high levels of productivity, quality, and reuse emanating from Japan may be a cause for

concern among some U.S. managers. At the least, Japanese software producers analyzed in this research have already set very high standards for performance, matching if not exceeding the best U.S. firms. If the Japanese continue or even improve upon these high levels, and master other elements needed to compete overseas in software development, companies from Japan may someday prove to be strong competitors in yet another industry.

Table 1: Negative Comments About Japanese Software Development

Comments:	Sources:	Quantitative Data:
PROCESS		
Copy/Rely on Western Tools and Techniques	Kim 1983	None
	Zelkowitz 1984	None
	Kishida 1987	None
	Cusumano 1989	Managers' Survey
Severe Shortage of Skilled Programmers	Lecht 1989	None
More Costly Software	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	U.S. Commerce 1984	None
	Gamaota & Frieman 1988	Anecdotal
GENERAL ASSESSMENT		
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

Table 6: Input and Output Metrics

Size	Means U.S. (median)	Japan	Wilcoxon Rank Sums Z Approximation
Work-years	102 (22.5)	47 (20.1)	.00
SLOC	343K (124.1)	433K (163.7)	.54
Fortran Equivalent	288K (77K) (83.9%	389K (144K) 89.8% of SLOC)	.62
Average Fortran Conversion	.90	.94	.38

Table 7: Personnel Experience

Average Years of Experience

	U.S.	Japan	Wilcoxon Rank Sums Z Approximation
Programmer	3.48	3.31	-.94
Designer	4.46	4.44	.35
Manager	7.61	7.00	.00

Table 8: Effort by Phase

Effort Distribution by Phase	U.S.	Japan	Wilcoxon Rank Sums Z Approximation
Design %	31	39	2.71***
Coding %	36	25	1.97**
Testing %	33	36	.66

Statistical Significance Levels:
** = .05, *** = .01

Table 9: Full-Time Effort Percentage by Phase

Full-Time vs. Part-Time %	U.S.	Japan	Wilcoxon Rank Sums Z Approximation
Full time design	82	77	- .38
Full time coding	81	81	.07
Full time testing	61	86	1.95**

Statistical Significance Levels:
** = .05

Table 10: Code Reuse

Code Reuse (% of Delivered Lines)	U.S. (median)	Japan	Wilcoxon Rank Sums Z Approximation
Code reuse	9.71 (3)	18.25 (11)	.71

Table 2: Positive Comments About Japanese Software Development

Comments:	Sources:	Quantitative Data:
PROCESS		
Ahead in Quality (defects) and "Product Engineering"	Zelkowitz 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
Ahead in Tool Usage	Kim 1983	None
	Zelkowitz 1984	Site Survey
	U.S. Commerce 1984	None
	Johnson 1985	None
	Gamaota & Frieman 1988	None
Ahead in 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
Ahead in Reuse	Standish 1984	Anecdotal
	Haavind 1986	Anecdotal
	Gamaota & Frieman 1988	1 Company's Data
	Tracz 1988	Anecdotal
	Cusumano 1989	Manager Survey
Ahead in Maintenance	Kishida 1986	None
	Nikkei/Cusumano 1988	User Surveys
Ahead in Project Management (Planning, Discipline, Teamwork)	Naur and Randall 1969	None
	Tajima & Matsubara 1981	1 Company's Data
	Zelkowitz 1984	None
	U.S. Commerce 1984	None
	Johnson 1985	None
	Belady 1986	None
PRODUCTS		
Good in Custom Programming	Nikkei/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

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

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	5 (21)	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

Table 5: Hardware Platforms[11]

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

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

Table 11: Tool Usage

Mean Tools/Methods Reported Used Per Project

	U.S. (median)	Japan	Wilcoxon Rank Sums Z Approximation
Number Used	4.04 (4)	4.13 (4)	-.06

Table 12: Detailed Tool Usage

Types of Tools Used - Percentage (%) Reporting Use

	U.S.	Japan
Analysis/Design		
Design support	29.2	31.3
Auto. Flowchart	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

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

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

Table 14: Independent Variables in Productivity Regression

Variable	Explanation
x_1	Dummy variable, =1 if application is data processing, else 0.
x_2	Dummy variable, =1 if hardware platform is mainframe, else 0.
x_3	Percentage of time in coding phase
x_4	Percentage of code reused

Table 15: Productivity Regression Results
(values of t-statistics shown in parentheses)

$$\underset{\text{SLOC/work-year}}{} = \underset{(2.31)}{10349} + \underset{(2.43)}{10107}\,x_1 + \underset{(-2.27)}{-7974}\,x_2 + \underset{(-1.66)}{-18133}\,x_3 + \underset{(4.99)}{443}\,x_4$$

$R^2 = .49$
Adj-$R^2 = .44$
F-stat = 8.572
n = 40

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	Size of system in 1000s of Fortran-equivalent SLOC
x_2	Percentage of time in testing phase
x_3	Number of testing/debugging tools used

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

$$\text{Failures/KSLOC} = \underset{}{9.65} + \underset{(2.84)}{.01\ x_1} - \underset{(3.90)}{15.44\ x_2} - \underset{(-.1.67)}{4.68\ x_3}\ \underset{(-1.84)}{}$$

$R^2 = .44$
Adj-R^2 = .38
F-stat = 7.080
n = 31

Appendix A: Sample Description

	Development Sites Receiving Forms	Number Returned	Number Discarded	Systems 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.

Appendix B: Companies and Product Areas Participating in the Study

U.S. Sites/Product Areas
Amdahl/Product Software
Amdahl/Engineering Software (2 Projects)
AT&T Bell Laboratories/Switching & Communications (2 Projects)
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 (2 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 Electronics 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 C: A Model of Software Reuse and Productivity

While it is generally agreed that reusability is a desirable concept, little work has been done on attempting to quantify the impact of reuse (Seppanen 1987). In fact, Standish has noted that "... it seems a great deal of insight could be gained from collecting and analyzing data derived from examples where software reuse techniques have been successfully applied in practice" (Standish 1984). One of the few economic models of reuse is provided by Gaffney and Durek (1988). They propose that the relative cost of a system that reuses code versus a completely new system is represented by the equation:

$$C = (1-R) + (R*b)$$

where C = the relative cost (=1 for an all new system)
R = the percentage of reused code
b = the relative cost of reuse versus writing new code (assumed to be <= 1 by Gaffney and Durek)

They further suggest that the value of b is related to the degree of reuse, i.e., whether just code is reused or whether there is related design reuse. They provide a hypothetical example, wherein the value of b is equal to $1-p$, where p is the percentage of time spent in the phase or phases represented by the reused component. For example, if coding is 15% of the life-cycle, then a system that only reuses code, but not requirements or design, would cost about 85% of the hypothetical cost of an all new system (Gaffney and Durek 1988: 48).

It is possible to test the Gaffney-Durek model using the U.S.-Japanese data-set. Given the value of R, the value of b can be estimated via linear regression, as follows:

$$y = B_0 + B_1 x_1 + B_2 x_2$$

where y = work years
x_1 = New KSLOC (= KSLOC*(1-R))
x_2 = Reused KSLOC (=KSLOC*R)

The estimate of this model for the data-set is as follows:

$$y = \underset{(-.44)}{-15.024} + \underset{(4.15)}{.306\, x_1} + \underset{(.55)}{.195\, x_2}$$

The signs of the coefficients of both the new and reused code are positive, as would be expected, because even reused code is not "free" (without effort). And, the coefficient on the reused code is less, only about two-thirds of that for the new code, which is also what would be expected, since if it took greater effort, presumably programmers would simply write new code (Woodfield et al. 1987). The estimated value of b then, is:

$$b = B_2 / B_1 = .64$$

Given an average percentage of time spent in the coding phase of 32% (.36*(24/40) + .25*(16/40)*100), the model would predict that b = (1-.32) = .68, quite close to that obtained from the estimate.

In terms of the sensitivity of this result, one concern is the low t-statistic on the estimate of B_2. This suggests that a high degree of

confidence cannot be placed in rejecting the null hypothesis that the "true" value of $B_2 = 0$. The noise in the estimate may be caused by differing levels of support for reuse, both in Japan and in the U.S. In other words, facilities with excellent support for reuse may show a lower number and those with poor support may show a higher number, leading to an average value of .195 with a large variance. However, lacking any other data, the value of .195 is the single best point estimate available. In addition, the fact that the value approximates that predicted from the theory also tends to increase confidence in the estimate. Of course, this result needs additional validation before any general claims can be made, but this initial result is encouraging.

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, No. 6, November 1983, pp. 639-648.

Abegglen, James C., and George Stalk, Jr., Kaisha: The Japanese Corporation, New York, Basic Books, 1985.

Arden, Bruce, What Can Be Automated?, Cambridge, MA, MIT Press, 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, pp. 160-175.

____, "A Model to Evaluate Variables Impacting Productivity on Software Maintenance Projects," Cambridge, MA, MIT Sloan School of Management Working Paper #2093-88, May 1989.

Belsley, D., E. Kuh and R. Welsch, Regression Diagnostics, New York, John Wiley and Sons, 1980.

Belady, Laszlo A., "The Japanese and Software: Is It a Good Match?" IEEE Computer, June 1986, pp. 57-61.

Boehm, Barry, "Improving Software Productivity," IEEE Computer, September 1987, pp. 43-57.

Boehm, B.W. Software Engineering Economics. Englewood Cliffs, N.J., Prentice-Hall, 1981.

Bradley, James V., Distribution-Free Statistical Tests, Englewood Cliffs, N.J., Prentice-Hall, 1968.

Brooks, Frederick P., The Mythical Man-Month, Reading, MA, Addison-Wesley, 1975.

____, "No Silver Bullet: Essence and Accidents of Software Engineering," IEEE Computer, April 1987, pp. 10-19.

Chrysler, E. "Some Basic Determinants of Computer Programming Productivity." Communications of the ACM, Vol. 21, No. 6, June 1978, pp. 472-483.

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

Conte, S., H. Dunsmore, and V. Shen, Software Engineering Metrics and Models, Reading, MA, Benjamin/Cummings, 1986.

Cusumano, Michael A., The Japanese Automobile Industry, Cambridge, MA, Harvard University Press, 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.

____, "The Software Factory: A Historical Interpretation," IEEE Software, March 1989, pp. 23-30.

Datamation, 1 June 1985, pp. 58-120.

Frank, Werner L., Critical Issues in Software, New York, John Wiley and Sons, 1983.

Freeman, P., ed., Tutorial: Software Reusability, Washington, D.C., IEEE Computer Society Press, 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, New York, John Wiley & Sons, 1982, V.2, pp. 177-185.

Gaffney, John E., Jr., and Thomas F. Durek, "Software Reuse - Key to Enhanced Productivity: Some Quantitative Models," ACM 27th Technical Symposium Proceedings, Gaithersburg, Maryland, June 1988, pp. 45-55.

Gamaota, George, and Wendy Frieman, Gaining Ground: Japan's Strides in Science and Technology, Cambridge, MA, Ballinger, 1988.

Haavind, Robert, "Tools for Compatibility," High Technology, August 1986, pp. 34-42.

Henderson, John, and Jay Cooprider, "Dimensions of I/S Planning and Design Technology," M.I.T. Center for Information Research, Working Paper #181, September 1988.

Hiyoshi, Yoshitaka, "Japan, Underdeveloped in Software, May be Changing," Software Magazine, November 1988, pp. 71-74.

Hunke, H. ed., Software Engineering Environments, Amsterdam, North-Holland, 1981.

Johnson, Colin, Electronic Engineering Times, "Software in Japan," 11 February 1985, p. 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, v. SE-10, n. 5, September, 1984, pp. 488-494.

____, Programming Productivity, New York, McGraw-Hill, 1986.

____, "A New Look at Languages," Computerworld, 7 November 1988, pp. 97-103.

Kemerer, Chris F., "An Empirical Validation of Software Cost Estimation Models," Communications of the ACM, v. 30, n. 5, May 1987, pp. 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, pp. 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, pp. 11-18.

Kriebel, C.H., and A. Raviv, "An Economics Approach to Modeling the Productivity of Computer Systems," Management Science, Vol. 26, No. 3, March 1980, pp. 297-311.

Lecht, Charles P., "Japanese Software No Threat," Computerworld, 8 May 1989, p. 21.

Matsumoto, Yoshiro, "A Software Factory: An Overall Approach to Software Production," in Peter Freeman, ed., Tutorial: Software Reusability, Washington, D.C., Institute of Electrical and Electronics Engineers, 1987, pp. 155-178.

McKeen, J.D. "Successful Development Strategies for Business Application Systems," MIS Quarterly, Vol. 7, No. 3, September 1983, pp. 47-65.

Naur, Peter, and Brian Randell, eds., Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Brussels, Scientific Affairs Division, NATO, January 1969.

Nikkei Computer, 14 March 1988, pp. 58-86, and 26 September 1988, pp. 66-99.

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

Pindyck, R. S., and Rubinfeld, D.L. Econometric Models and Economic Forecasts, New York, McGraw-Hill Book Company, 1981.

Putnam, L. H. "General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Transactions on Software Engineering, Vol. 4, 1978, pp. 345-361.

Ramamoorthy, C.V., et al., "Software Engineering: Problems and Perspectives," IEEE Computer, October 1984, pp. 191-209.

Rifkin, Glenn, and J.A. Savage, "Is U.S. Ready for Japan Software Push," Computerworld, 8 May 1989, p. 1.

Sakai, Toshio, "Software: The New Driving Force," Business Week, 27 February 1984, pp. 96-97.

Schonberger, Richard J., Japanese Manufacturing Techniques, New York, The Free Press, 1982.

Selby, Richard, "Empirically Analyzing Software Reuse in a Production Environment," 1988.

Seppanen, Veikko, "Reusability in Software Engineering," in Peter Freeman, ed., Tutorial: Software Reusability, Washington, D.C., IEEE Computer Society Press, 1987, pp. 286-297.

Stabell, C. B. "Office Productivity: A Macroeconomic Framework for Empirical Research," Office Technology and People, Vol. 1, No. 1, 1982, pp. 91-106.

Standish, Thomas A., "An Essay on Software Reuse," IEEE Transactions on Software Engineering, v. SE-10, n. 5, September 1984, pp. 494-497.

Tracz, Will, "Software Reuse Myths," Software Engineering Notes, v. 13, n. 1, January 1988, pp. 17-21.

Tajima, Denji, and Tomoo Matsubara, "The Computer Software Industry in Japan," IEEE Computer, May 1981, pp. 89-96.

____, "Inside the Japanese Software Industry," IEEE Computer, March 1984, pp. 34-43.

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

U.S. Department of Labor, Bureau of Labor Statistics, Monthly Labor Review, Washington, D.C., monthly issues, 1989.

Uttal, Bro, "Japan's Persistent Software Gap," Fortune, 15 October 1984, pp. 151-160.

Vogel, Ezra F., Japan as Number 1: Lessons for America, Cambridge, MA, Harvard University Press, 1979.

Walston, C. E., and C.P. Felix, "A Method of Programming Measurement and Estimation," IBM Systems Journal, Vol. 16, No. 1, 1977, pp. 54-73.

Woodfield, S., D. Embley, and D. Scott," Can Programmers Reuse Software?" IEEE Software, July 1987, pp. 52-59.

Zelkowitz, Marvin V., et al., "Software Engineering Practices in the U.S. and Japan," IEEE Computer, June 1984, pp. 57-66.