12-31-1994

Project Size and Software Maintenance Productivity: Empirical Evidence on Economies of Scale in Software Maintenance

Rajiv Banker
University of Minnesota

Sandra Slaughter
University of Maribor

Paula Swatman
Monash University

Rene Wagenaar
Erasmus University

Clive Wrigley
Erasmus University and McGill University

PROJECT SIZE AND SOFTWARE MAINTENANCE PRODUCTIVITY: EMPIRICAL EVIDENCE ON ECONOMIES OF SCALE IN SOFTWARE MAINTENANCE

Rajiv D. Banker
Sandra A. Slaughter
Carlson School of Management
University of Minnesota

ABSTRACT

Although appropriate sizing of software projects is a concern in software *development*, Information Systems managers generally do not consider *maintenance* project size as a potential influence on software maintenance productivity. This view ignores potential efficiency gains which may arise from proactive renovation strategies such as batching similar maintenance requests into larger projects. In this study, we explore the relationship between project size and productivity for software maintenance projects at a major national mass merchandising retailer. Using a non-parametric methodology called Data Envelopment Analysis (DEA) for estimating the functional relationship between maintenance inputs and outputs, we determine the most productive scale size for a set of maintenance projects at this organization. In addition, we also employ DEA-based heuristics to test for the existence of returns to scale for the projects. Our results indicate the presence of significant scale economies in these software maintenance projects. The most productive scale size is larger than 90% of the projects included in our sample. These results imply that there may be potential to increase productivity in software maintenance at this organization by grouping smaller modification projects into larger planned releases.

1. INTRODUCTION

Software maintenance claims a significant and growing portion of Information Systems (IS) resources, requiring from 50% to 80% of the IS budget (Gallant 1986; Freedman 1986). Yet, perhaps because software maintenance tends to be viewed as a necessary evil, many technological and process improvements are directed toward software development. There is a growing realization, however, that productivity improvements in software maintenance can enable re-deployment of IS resources to other activities (Arthur 1988). Thus, there is increased motivation to more effectively manage software maintenance.

Typically, IS managers do not consider maintenance project *size* as a potential influence on software maintenance productivity (Swanson and Beath 1989; Martin and McClure 1983). By far, the bulk of the literature on project size planning for information systems is devoted to planning for development projects (e.g., Humphrey 1989; DeMarco 1982). In the maintenance area, however, much of the work arises as small, imprecise, ambiguous requests of unpredictable urgency from the users for modifications to existing software systems. Many organizations handle these requests by simply processing them as they are submitted.[1] This approach ignores benefits which may arise from proactive renovation strategies such as a release control concept (Branch, et al. 1985; NcNeil 1979) in which user requests for modifications to installed applications are grouped into a package and implemented in a release.

Although there has been considerable interest in investigating project scale economies for software development (e.g., Byrnes, Frazier and Gulledge 1993; Banker and Kemerer 1989), very little examination of this issue has been done for software maintenance projects. In this study, we explore the relationship between maintenance project size and productivity for twenty-seven software maintenance projects completed over a two year timeframe at a major mass merchandising retailer. In the context of our study, we define a maintenance project to include the

activities necessary to make and implement the software changes for a particular user request. Using the non-parametric Data Envelopment Analysis (DEA) methodology for estimating the functional relationship between maintenance project inputs and outputs, we estimate the most productive scale size for this set of maintenance projects. In addition, we also employ DEA-based heuristics to test for returns to scale (Banker and Chang 1994). Our results indicate that significant scale economies are present in the software maintenance projects at our research site.

In the following sections of the paper, we apply concepts of productivity from microeconomics to software maintenance, outline our research methodology, and present an analysis of our empirical data. We conclude with a discussion of the implications of our results for software maintenance management and future research.

2. AN ECONOMIC VIEW OF SOFTWARE MAINTENANCE PRODUCTIVITY

2.1 Productivity Concepts from Production Economics

Several concepts from production economics are relevant to the analysis of productivity. The *production process* defines the technical means by which inputs (materials and services) are combined to produce outputs (goods or services). This technical relationship is represented by the *production function* which expresses the maximum level of outputs produced for each given level of inputs (Varian 1992). Production economics emphasizes the *frontier* or best practice notion for a production function, with deviations from the frontier reflecting inefficiencies in individual observations (Aigner and Chu 1968; Banker 1993).

An important characteristic of the production process is *returns to scale*. Returns to scale is defined as the relative increase in output as all inputs are increased proportionately so that the relative factor mix does not change (Varian 1992). A production process exhibits *constant returns to scale* if, when all inputs are increased by a given proportion k, output increases by the same proportion. If output increases by a proportion greater than k, there are *increasing returns to scale*; and if output increases by a proportion smaller than k, there are *decreasing returns to scale*.

There is a direct relationship between returns to scale and productivity of the production process. Maximum average productivity is achieved when there are constant returns to scale. Local *economies of scale* are present where average productivity is increasing and *diseconomies of scale* occur where average productivity is decreasing.[2] For a single-input, single-output case, the *most productive scale size* (MPSS) for the production process is the scale which maximizes average productivity; this occurs at the point where there are local constant returns to scale. At the MPSS, all productivity gains due to increasing returns have been exploited, but decreasing returns have not yet set in. A relatively low MPSS indicates that decreasing returns set in early, while a high MPSS suggests that increasing returns prevail for larger scale sizes. Figure 1 provides a graphical illustration of these concepts.

2.2 Application of Productivity Concepts to Software Maintenance

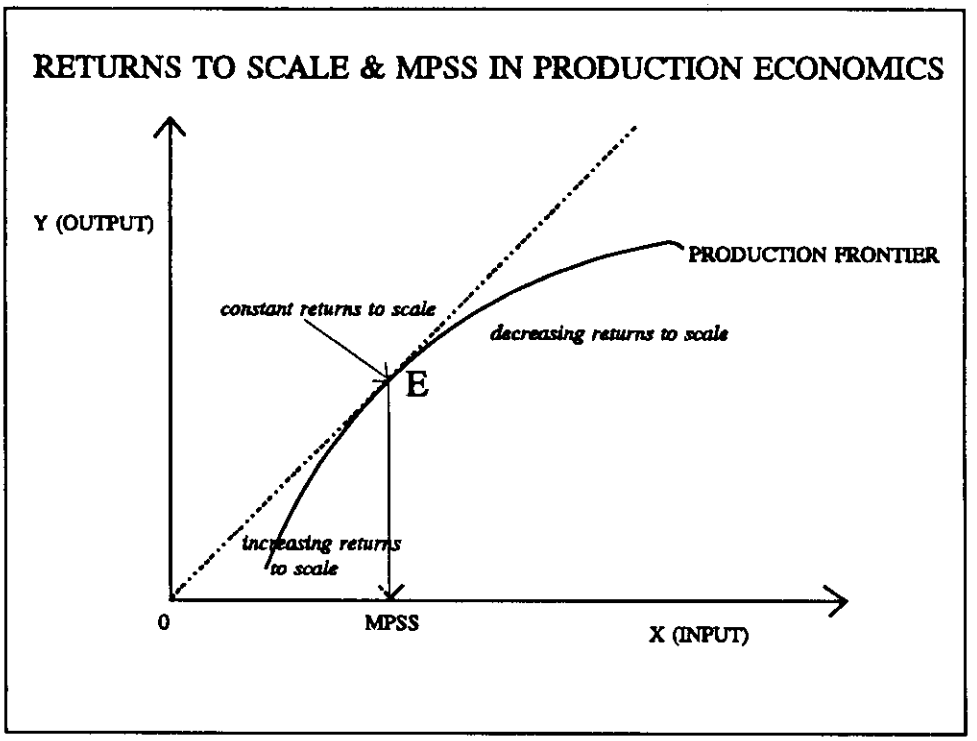
These concepts from production economics provide insight into the relationship between software maintenance project size and productivity. Software maintenance is typically viewed as the modification of a software product after delivery to correct faults, improve performance or adapt to a changed environment (Swanson 1976; Schneidewind 1987). From an economic perspective, software maintenance can be conceptualized as a production process (Banker, Datar, and Kemerer 1991). In this sense, software maintenance projects convert inputs (the effort of maintenance professionals) into outputs (modified software). The maintenance production function represents the optimal relationship between maintenance effort and modified software for maintenance projects. This implies that software maintenance projects which deviate from the production frontier are less efficient than those which lie along the frontier.

The most productive maintenance project size occurs where proportionate increases in team effort result in the same proportionate increase in modified software. If, for example, at a certain project scale, a doubling of maintenance effort results in three times the amount of modified software, there are increasing returns to scale, and average productivity increases by increasing project scale; if the reverse is true, then the maintenance project exhibits decreasing returns to scale, and average productivity increases by decreasing project scale. If a doubling of maintenance effort results in double the amount of modified software, there are constant returns to scale and maximum maintenance productivity.

2.3 Scale Economies in Software Maintenance

We propose that economies of scale are present in software *maintenance* as well as in software development. There have been several studies of scale economies in software development (Walston and Felix 1977; Jeffery and Lawrence 1979; Vessey 1986; Banker and Kemerer 1989; Byrnes, Frazier and Gulledge 1993; Banker, Chang and

Figure 1



Kemerer 1994). An empirical analysis of returns to scale in software *development* for eight publicly available data sets demonstrated the presence of both scale economies and diseconomies (Banker and Kemerer 1989). In general, this study found that smaller development projects are characterized by increasing returns, while larger projects are characterized by diminishing returns to scale. The authors suggest that, in software development, productivity increases on larger projects arise from spreading fixed project management overhead over a larger base, and from greater use of specialized personnel and tools (Boehm 1981). However, eventually, larger project size tends to increase the complexity of interface requirements, the number of inter- and intra-project communication paths, and the requirements for documentation (Conte, Dunsmore and Shen 1986; Brooks 1975). Thus, average productivity of the project team is likely to decline beyond the most productive scale size of the project.

Although there are similarities between software development and maintenance, software maintenance is different because the programming team must spend a significant amount of time attempting to understand the purpose and construction of the programs to be modified (Littman, et al. 1987; Fjeldstad and Hamlen 1983). Thus, while economies and diseconomies of scale may arise for similar reasons as in software development, there may be additional sources contributing to scale economies in software maintenance. For example, efficiency gains in maintenance may occur where requests for a particular system are batched, so that the maintenance team can spread learning curve effects from becoming familiar with that system over several requests. In addition, it may be more productive to make several changes to a system and then perform a single system test prior to re-installation instead of a series of tests. There may also be efficiencies realized from accomplishing and implementing documentation updates from a batch of changes, rather than continually modifying documentation as small changes are made.

Although scale economies in software maintenance may be present, there may be organizational resistance against a proactive release control process. The nature of maintenance project requests tends to differ from software development in that maintenance requests are generally smaller, incremental to existing systems, unpredictably urgent, and have a more immediate impact on the user's work. Thus, although "true" emergency repairs account for only a small portion of maintenance work (Lientz and Swanson 1980), there may be institutional pressures to complete all maintenance requests upon demand. In describing the implementation of an SRD (System Release Discipline) approach to maintenance, McNeil (1979) notes some of the difficulties encountered:

> [S]ome old habits may die hard. Software staff may feel they are giving up control of "their" system, and indeed they must sacrifice the freedom to bomb it at will. Users, for their part, may object to being unable to demand a new feature for the system on an overnight basis. Their payoff must wait, but at least when the feature is implemented it is more likely to work properly. Operations staff may have been getting blamed for every system problem anyhow, so they can be expected to favor any change in procedures which makes their lives less hectic. [p. 112]

These sentiments are emphasized by software maintenance professionals participating in a study by Dekleva (1992):

> [D]ue to the length of maintenance tasks, new requests frequently come along before the ongoing task is completed. There is always something more critical or another user with a problem. A great deal of time is wasted on stopping and starting maintenance tasks. [p. 17]

Thus, to justify change in maintenance practice, it is critical to provide evidence of the potential benefits to be gained from managerial innovations which take advantage of scale economies in accomplishing software maintenance tasks. Our study makes an initial contribution in this area by determining the existence and extent of scale economies in software maintenance projects at our research site.

3. METHODOLOGY

3.1 Research Site

Data for this study were collected at a national mass merchandising retailer. The IS department for the organization is located at company headquarters and supports all centralized computer processing activities for the company. In 1983, the IS department was divided into separate development and maintenance teams, with the development team working exclusively on development of new systems and major enhancements, and the maintenance team responsible for support of existing systems and minor enhancements. The organization has a large investment in computer software in COBOL, written in the late 1970s and the 1980s, and running on large IBM mainframe computers. At the end of 1992, the size of the application portfolio was estimated at 85,000 function points (approximately 14 million lines of code).

On average, a typical maintenance programmer supports 2,200 function points. Small (one to three person) teams are responsible for supporting major applications. Types of maintenance work include user support (non-programming activities such as responding to user queries), repairs (corrections of application defects), and enhancements (addition or modification of application functionality). The maintenance team does not utilize a formal change management program, i.e., maintenance requests for all types of work are processed upon receipt and as time permits. That the current approach to managing maintenance projects may be unsatisfactory is reflected in the remarks of one of the maintenance managers at the site:

> It would be nice if I could plan requests, if I had extended time to plan and schedule. For example...there are so many small projects in the XXX Area which I'd like to batch if I could. I'd like to use a release method (like software vendors)...but the users wouldn't go for it. It's too dynamic here. We're not disciplined enough to plan ahead. [Interview Transcript, January 25, 1993]

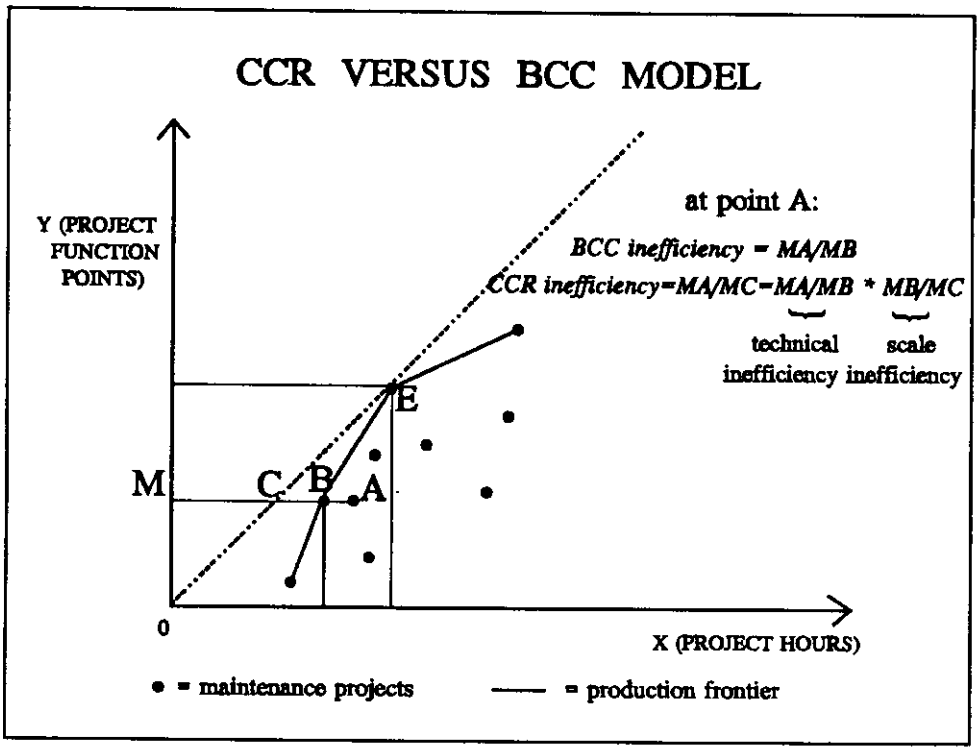
3.2 Data Collection

Our general strategy was to collect data retrospectively for completed maintenance projects. We obtained data for twenty-seven software maintenance projects which modified twelve different application systems. Each project modified only one application; thus, in some cases, there were multiple projects for the same application. To control for the influence of extraneous factors, three criteria were used to select projects for inclusion in the study: recency of completion, similarity of project type, and similarity of programming language. Project recency is important because personnel turnover and lack of documentation retention make accurate data collection impossible for older projects. In addition, technology and personnel involved are more similar for projects completed in a shorter and more recent timeframe; this enables cross-project comparisons. Similarity of project type is important because this improves homogeneity of the projects analyzed. We included only projects which modified functionality for the application systems; thus, projects such as package installations or conversions to new operating systems were not compared with projects which modify functionality. Our final criterion was similarity of programming language. All projects included modifications to COBOL programs. Therefore, the results of our analysis are not confounded by the effects of multiple programming languages. After discussions with IS staff at the research site, only projects completed within a two-year timeframe between January, 1991, and December, 1992, were considered for inclusion in the study. Of the forty projects initially considered for inclusion, thirteen were eliminated because they were either non-COBOL or non-modification projects. Data were collected in early 1993. Table 1 presents a profile of the twenty-seven projects included in this study.

Table 1. Profile of Projects Included with Efficiency Ratings

Project#	Project Hours	Project Function Points	Efficiency Rating (CCR)	Efficiency Rating (BCC)
1	48.0	13.0	0.2019	1.0000
2	85.0	8.0	0.0702	0.5647
3	99.0	12.0	0.0904	0.4848
4	921.0	304.0	0.2461	0.2499
5	198.0	49.0	0.1845	0.3563
6	274.0	86.0	0.2340	0.3420
7	515.0	120.0	0.1737	0.2233
8	265.0	150.0	0.4220	0.5048
9	632.0	282.0	0.3326	0.3425
10	76.8	10.0	0.0971	0.6250
11	397.5	108.0	0.2026	0.2704
12	104.0	8.0	0.0573	0.4615
13	222.0	136.0	0.4567	0.5632
14	249.0	334.0	1.0000	1.0000
15	364.5	97.0	0.1984	0.2760
16	2450.6	410.0	0.1247	0.1632
17	69.0	17.0	0.1837	0.7319
18	49.5	11.0	0.1657	0.9697
19	145.5	13.0	0.0666	0.3299
20	370.0	42.0	0.0846	0.1788
21	1675.0	1052.0	0.4682	1.0000
22	141.0	10.0	0.0529	0.3404
23	61.0	17.0	0.2078	0.8279
24	71.0	19.0	0.1995	0.7289
25	460.0	187.0	0.3031	0.3412
26	397.0	57.0	0.1071	0.1903
27	927.0	247.0	0.1986	0.2098
* Mean *	417.3	140.7	0.2270	0.4917
* Std Dev *	543.0	216.0	0.1894	0.2720

Figure 2



3.3 DEA Methodology

To investigate the relationship between maintenance project size and productivity, we employ Data Envelopment Analysis (DEA), a non-parametric methodology for production frontier estimation developed by Charnes, Cooper, and Rhodes (1981) and extended to a formal production economics framework by Banker, Charnes, and Cooper (1984). DEA is a technique that employs linear programming to map a production frontier for input and output data. In the context of software maintenance, DEA evaluates the relative productivity of each maintenance project by comparing it against a composite project that is constructed as a convex combination of other projects in the data set.

The methodology of Charnes, Cooper and Rhodes (the "CCR" model) enables the estimation of aggregate technical and scale inefficiencies in the software maintenance projects, while the methodology of Banker, Charnes and Cooper (the "BCC" model) facilitates the estimation of the technical inefficiency of a project at the given scale of production. Since the CCR model estimates both scale and technical inefficiencies, it is an appropriate measure of productivity when maintenance project size is discretionary, i.e., under the control of the IS manager. The CCR model identifies the project which lies at the point of constant returns to scale to be most productive since maximum average productivity is maximized at this project size. When maintenance project size is "fixed," i.e., not under the control of the IS manager, the BCC model provides a measure of technical efficiency for a given project size. Thus, in the BCC model, any project along the production frontier for the data set is considered to be efficient since technically efficient points lie at the frontier. Figure 2 illustrates the differences between the BCC and CCR models. In this example, points A, B, and E represent three software maintenance projects. The CCR model identifies project E as the most productive, given that project scale size can be varied at the discretion of the IS manager. The BCC model identifies projects B and E as technically efficient, because they represent the highest productivity when scale size is given and cannot be altered by the IS manager. The technical efficiency of project A is determined relative to project B which lies along the BCC production frontier, and the scale efficiency of project A is determined relative to point C (not representing a maintenance project) which lies along the CCR frontier line connecting the origin to project E (the most productive scale size). In our study, we estimate both CCR and BCC models because tests of returns to scale in DEA use both models to construct the test statistics (Banker and Chang 1994).

We employ DEA rather than a parametric model to estimate the production relationship between maintenance input and output. Since DEA does not impose a specific form on the production function and maintains relatively few assumptions,[3] its estimates are likely to be more robust than those obtained from parametric models that postulate a certain structure such as a linear or quadratic form for the software maintenance production function (Banker and Maindiratta 1988).

Using DEA, we estimate the production function relating maintenance project input (project team labor) to output (modified software functionality). We focus on labor hours as the critical input of interest, because IS personnel staff time is the most expensive and scarce resource in software development and maintenance, accounting for over half of the IS budget (Grammas and Klein 1985). Project team labor is measured by the total number of labor hours logged to the project by the maintenance team and is obtained from the organization's project time tracking system. Size of modified software functionality is assessed by the number of function points added, changed or deleted by the project. Function points have been demonstrated to be a reliable estimate of software size (Kemerer 1993). A function point corresponds to an end-user business function such as *sales order entry*. The functions are organized into five groups which reflect how the function is accomplished in the software: external inputs, external outputs, external inquiries, logical internal files and external interface files (Albrecht and Gaffney 1983). Thus, function points provide an estimate of the number of files, reports and screens updated by the maintenance project. For this study, the function point figures are obtained from the organization's Quality Assurance Department which is responsible for counting of projects and applications.

We also use DEA to estimate the MPSS for the maintenance projects. Since we are interested in a single input-single output production correspondence, the computation is relatively straightforward. The MPSS is given by the project size for which the ratio of project size to labor hours (or the average productivity) is the largest for all of the observations (Banker 1984; Banker and Kemerer 1989). Projects which are larger (smaller) than the MPSS correspond to decreasing (increasing) returns to scale.

Finally, we use new DEA-based heuristics (Banker and Chang 1994) to formally test for returns to scale. There are two heuristics, depending on whether the deviations of observed data from the estimated production function are postulated to be distributed as exponential or half-normal. As a sensitivity check, we construct test statistics under both assumed forms of distribution for the deviations. These tests indicate whether the observations can be described well with a constant returns to scale model, with only an increasing (or decreasing) returns to scale model, or with a model that allows for the presence of both increasing and decreasing returns to scale in different data ranges. To construct the test statistics, we estimate both the CCR and BCC models within DEA.

4. EMPIRICAL RESULTS

4.1 BCC and CCR Efficiency Ratings for the Maintenance Projects

To assess the pure technical efficiency $e^{BCC} = 1/\Theta(Y_0,X_0)$ of each maintenance project (Y_0,X_0), we estimate the following BCC model:

$$(4.a)\quad \Theta^B(Y_0,X_0) \equiv \text{Max } \Theta$$

subject to

$$(4.b)\quad \sum_{j=1}^{27} \lambda_j Y_j - Y_0/\Theta \leq 0$$

$$(4.c)\quad \sum_{j=1}^{27} \lambda_j X_j \geq X_0$$

$$(4.d)\quad \sum_{j=1}^{27} \lambda_j = 1$$

$$(4.3)\quad \Theta \text{ and } \lambda_j \geq 0$$

where
- j = project observation number
- Y = project Function Points
- X = project Labor Hours
- Θ^B = inefficiency variable, BCC model
- λ = weight

This model is solved as a linear program in the efficiency variable $e^{BCC} = 1/\Theta$ and the weights λ_j. As shown by Banker (1993), the DEA estimator of Θ is statistically consistent, and the asymptotic empirical distribution of the DEA estimates retrieves the true distribution of Θ under the maintained assumptions embodied in the DEA postulates of convexity, monotonicity, envelopment and likelihood of efficient performance. These assumptions are consistent with both increasing and decreasing returns to scale and do not impose constant returns to scale. The efficiency ratings ($e^{BCC} \leq 1.0000$) for the twenty-seven software maintenance projects estimated under the BCC model are presented in the fifth column of Table 1. Projects with a rating of 1.0000 are the most efficient given their scale size; the

further the rating is from 1.0000, the less efficient the project. For our sample of maintenance projects, three projects are identified as efficient using the BCC model.

Estimates of aggregate technical and scale efficiency under the CCR model, $e^{CCR} = 1/\Theta$, are obtained by solving the above linear program, except that the objective function in (4.a) is maximized subject only to constraints (4.b), (4.c), and (4.e). The CCR estimator is also statistically consistent under the maintained DEA assumptions, with the addition of a postulate for constant returns to scale. We refer to the CCR inefficiency estimate as Θ^C. The efficiency ratings ($e^{CCR} \leq 1.0000$) for the twenty-seven software maintenance projects estimated under the CCR model are presented in the fourth column of Table 1. The project with a rating of 1.0000 is the most productive; the further the rating is from 1.0000, the less is the average productivity of the project.

4.2 Most Productive Scale Size

In Figure 1, the MPSS occurs at the point E, which reflects the maximum observed average productivity across all observations (expressed as the ratio of project function points to labor hours). For this sample, $E \approx 1.34$ function points per hour. The project size at which E is achieved is for project #14 at 334 function points and 249 labor hours. Only two of the twenty-seven maintenance projects are larger than this scale size and exhibit decreasing returns to scale, implying that 90% of the projects are characterized by increasing returns to scale, i.e., they are too small to attain the maximum average productivity.

4.3 Tests for Returns to Scale

We construct heuristics to test whether there are non-constant returns to scale for this set of maintenance projects. Since the DEA estimator is statistically consistent, under the null hypothesis of constant returns to scale, the asymptotic empirical distributions of the DEA estimates of Θ^B and Θ^C are identical, each recovering the true distribution of Θ (Banker 1993). This motivates the development of two heuristics for semiparametric statistical tests of constant, non-increasing, and non-decreasing returns to scale (Banker and Chang 1994). The two heuristics correspond to different maintained assumptions about the distribution (exponential versus half-normal) of the inefficiency variable Θ. The robustness of these DEA based heuristics has been demonstrated for different underlying production functions and inefficiency distributions as well as finite sample sizes (Banker and Chang 1994).

Table 2 summarizes the test statistics, the hypotheses and results. Under both heuristics, the null hypothesis of constant returns to scale is rejected at the 5% level of significance. This suggests that variable returns to scale are present in the data set. We reject the null hypotheses of decreasing returns to scale and non-increasing returns to scale at the 5% level of significance under both heuristics. Given this result, we conclude that this data set is characterized primarily by increasing returns to scale. Thus, our tests of returns to scale indicate that the maintenance projects are largely characterized by increasing returns to scale and are too small for maximum productivity. To assess the validity and robustness of our results, we conducted a number of sensitivity analyses. Following the intuition of Richmond (1974), we iteratively removed efficient projects from our data set and re-computed our test statistics to assess the extent of the influence of efficient projects on our results.[4] In all cases, our analysis confirmed the robustness of our result that maintenance projects at our research site are characterized by increasing returns to scale, and most of them are too small for maximum productivity.

5. DISCUSSION

In this paper, we explore the relationship between maintenance project size and productivity for software maintenance projects in a field setting. Using the DEA methodology for estimating the functional relationship between maintenance project size and labor hours, we estimate the most productive scale size for this set of maintenance projects. In addition, we employ DEA-based heuristics to examine returns to scale for the projects. Our results indicate the presence of significant scale economies in these software maintenance projects. The most productive scale size for these projects is larger than most of the projects included in our sample.

Our results provide evidence that scale economies are present in software maintenance at our research site. The presence or absence of scale economies at a given maintenance project size is important for software maintenance management because of the influence on maintenance productivity. Since the projects in our study are relatively large in size for maintenance efforts (with an average of over 400 hours), they provide a conservative test of economies of scale, implying that there may be significant potential to increase productivity by grouping typical "quick fix" maintenance projects into larger planned releases. This information can serve as justification for implementing a proactive change management program in which managers scale future projects accordingly so as to maximize the productivity of software maintenance effort. Information on scale economies can also be used within an existing change management program to assist in grouping projects such that maintenance productivity is maximized.

Table 2. Returns to Scale Tests

DEA heuristics based on exponential distribution

Null Hyp.	Alt. Hyp.	Test Statistic	F Statistic	Critical $F_{.95}(27,27)$	Result
CRS	VRS	$\sum_{j=1}^{27}(\Theta_j^C-1)/\sum_{j=1}^{27}(\Theta_j^B-1)$	3.45696	1.53	Reject CRS
NDRS	DRS	$\sum_{j=1}^{27}(\Theta_j^C-1)/\sum_{j-1}^{27}(\Theta_j^D-1)$	1.01879	1.53	Fail to Reject NDRS
NIRS	IRS	$\sum_{j=1}^{27}(\Theta_j^C-1)/\sum_{j=1}^{27}(\Theta_j^E-1)$	3.24964	1.53	Reject NIRS

DEA heuristics based on half-normal distribution

Null Hyp.	Alt. Hyp.	Test Statistic	F Statistic	Critical $F_{.95}(27,27)$	Result
CRS	VRS	$\sum_{j=1}^{27}(\Theta_j^C-1)^2/\sum_{j=1}^{27}(\Theta_j^B-1)^2$	11.24387	1.65	Reject CRS
NDRS	DRS	$\sum_{j=1}^{27}(\Theta_j^C-1)^2/\sum_{j-1}^{27}(\Theta_j^D-1)^2$	1.17010	1.65	Fail to Reject NDRS
NIRS	IRS	$\sum_{j=1}^{27}(\Theta_j^C-1)^2/\sum_{j=1}^{27}(\Theta_j^E-1)^2$	9.60929	1.65	Reject NIRS

Key: CRS = constant returns to scale
VRS = variable returns to scale
NDRS = non-decreasing returns to scale
DRS = decreasing returns to scale
NIRS = non-increasing returns to scale
IRS = increasing returns to scale

Θ^C = CCR inefficiency variable
Θ^B = BCC inefficiency variable
Θ^D = NDRS inefficiency variable
Θ^E = NIRS inefficiency variable

In addition, our findings have interesting implications for software maintenance research. They suggest that researchers interested in software productivity should consider project size as an important influence on productivity in software maintenance as well as in software development. There are several possible extensions to this study. Research in software development indicates that the MPSS varies widely across organizational environments (Banker and Kemerer 1989). Thus, future work could assess whether this finding is replicated for software maintenance and could try to identify factors that contribute to some organizations' ability to successfully manage larger maintenance projects. A study could also assess the effects on maintenance productivity of other scale-related factors, such as the size of the project team and calendar length of the project. A final possibility is to use the DEA methodology to assess the performance of an organization which utilizes a proactive change management program in software maintenance.

6. REFERENCES

Aigner, D. J., and Chu, S. F. "On Estimating the Industry Production Function." *American Economic Review*, Volume 58, 1968, pp. 826-839.

Albrecht, A. J., and Gaffney, J. "Software Function, Source Lines of Code, and Development Effort Prediction: a Software Science Validation." *IEEE Transactions on Software Engineering*, Volume SE-9, Number 6, 1983, pp. 639-648.

Arthur, L. J. *Software Evolution.* New York: John Wiley & Sons, Inc., 1988.

Banker, R. D. "Estimating Most Productive Scale Size Using Data Envelopment Analysis." *European Journal of Operations Research*, Volume 17, Number 1, July 1984, pp. 35-44.

Banker, R. D. "Maximum Likelihood, Consistency and Data Envelopment Analysis: A Statistical Foundation." *Management Science*, Volume 39, Number 10, October 1993.

Banker, R. D., and Chang, H. "Tests of Returns to Scale in Data Envelopment Analysis." Forthcoming in *International Journal of Productivity*, 1994.

Banker, R. D.; Chang, H.; and Kemerer, C. F. "Evidence on Economies of Scale in Software Development." Forthcoming in *Information and Software Technology*, 1994.

Banker, R. D.; Charnes, A.; and Cooper, W. W. "Some Models for Estimating Technical and Scale Inefficiencies in DEA." *Management Science*, Volume 30, Number 9, September 1984, pp. 1078-1092.

Banker, R. D.; Datar, S.; and Kemerer, C. F. "A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects." *Management Science*, Volume 37, Number 1, January 1991, pp. 1-18.

Banker, R. D., and Kemerer, C. F. "Scale Economies in New Software Development." *IEEE Transactions on Software Engineering*, Volume 15, Number 10, October 1989, pp. 1199-1205.

Banker, R. D., and Maindiratta, A. "Nonparametric Analysis and Allocative Efficiencies in Production." *Econometrica*, Volume 56, Number 6, November 1988, pp. 1315-1332.

Boehm, B. W. *Software Engineering Economics*. Englewood Cliffs, New Jersey: Prentice-Hall, 1981.

Branch, M. A.; Jackson, M. C.; Laviolette, M. C.; and Frankel, E. C. "Software Maintenance Management." *Conference on Software Maintenance*, 1985, pp. 62-68.

Brooks, F. P. *The Mythical Man Month*. Reading, Massachusetts: Addison-Wesley, 1975.

Byrnes, P. E.; Frazier, T. P.; and Gulledge, T. R. "Returns-to-Scale in Software Production: A Comparison of Approaches." In T. R. Gulledge and W. P. Hutz (Editors), *Analytical Methods in Software Engineering Economics*. New York: Springer-Verlag, 1993, pp. 75-97.

Charnes, A.; Cooper, W. W.; and Rhodes, E. "Evaluating Program and Managerial Efficiency: An Application of Data Envelopment Analysis to Program Follow Through." *Management Science*, Volume 27, Number 6, June 1981, pp. 668-697.

Conte, S.; Dunsmore, H.; and Shen, V. *Software Engineering Metrics and Models*. Reading, Massachusetts: Benjamin Cummings, 1986.

Dekleva, S. "Delphi Study of Software Maintenance Problems." *Conference on Software Maintenance*, 1992, pp. 10-17.

DeMarco, T. *Controlling Software Projects*. New York: Yourdon Press, 1982.

Fjeldstad, R. K., and Hamlen, W. T. "Application Program Maintenance Study: Report to Our Respondents." In G. Parikh and H. Zvegintzov (Editors), *Tutorial on Software Maintenance*. Silver Spring, Maryland: IEEE Computer Society Press, 1983.

Freedman, D. H. "Programming without Tears." *High Technology*, Volume 6, Number 4, 1986, pp. 38-45.

Gallant, J. "Survey Finds Maintenance Problem Still Escalating." *Computerworld*, Number 20, January 27, 1986.

Grammas, G. W., and Klein, J. R. "Software Productivity as a Strategic Variable." *Interfaces*, Volume 15, Number 3, May-June 1985, pp. 116-126.

Humphrey, W. S. *Managing the Software Process*. Reading, Massachusetts: Addison-Wesley, 1989.

Jeffery, D. R., and Lawrence, M. J. "An Inter-Organisational Comparison of Programming Productivity." In *Proceedings of the Fourth International Conference on Software Engineering*, 1979, pp. 369-377.

Kemerer, C. F. "Reliability of Function Points Measurement." *Communications of the ACM*, Volume 36, February 1993, pp. 85-97.

Lientz, B. P., and Swanson, E. B. *Software Maintenance Management*. Reading, Massachusetts: Addison-Wesley, 1980.

Littman, D. C.; Pinto, J.; Letovsky, S.; and Soloway, E. "Mental Models and Software Maintenance." *Journal of Systems and Software*, Volume 7, 1987, pp. 341-355.

Martin, J., and McClure, C. *Software Maintenance: The Problem and its Solution*. Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

McNeil, D. H. "Adopting a System Release Discipline." *Datamation*, January 1979, pp. 110-117.

Nosek, J. T., and Palvia, P. "Software Maintenance Management: Changes in the Last Decade." *Journal of Software Maintenance*, Volume 2, Number 3, September 1990, pp. 157-174.

Richmond, J. "Estimating the Efficiency of Production." *International Economic Review*, Volume 15, June 1974, pp. 515-521.

Schneidewind, N. F. "The State of Software Maintenance." *IEEE Transactions on Software Engineering*, Volume SE-13, Number 3, March 1987, pp. 303-310.

Swanson, E. B. "The Dimensions of Maintenance." *Proceedings of the Second International Conference on Software Engineering*, San Francisco, California, October 13-15, 1976, pp. 492-496.

Swanson, E. B., and Beath, C. M. *Maintaining Information Systems in Organizations*. New York: John Wiley and Sons, 1989.

Varian, H. R. *Microeconomic Analysis*. New York: Norton, 1992.

Vessey, I. "On Program Development Effort and Productivity." *Information and Management*, Volume 10, 1986, pp. 255-266.

Walston, C. E., and Felix, C. P. "A Method of Programming Measurement and Estimation." *IBM Systems Journal*, Volume 16, Number 1, 1977, pp. 54-73.

7. ENDNOTES

1. A survey of software maintenance practices by Nosek and Palvia (1990, p. 170), for example, found that while 89% of the responding organizations reported *logging* user requests for changes, only 33% reported *batching* change requests.

2. In production economics, economies of scale are defined at specific volume levels in a production process and should be described as *local*. In dealing with single input-single output production correspondences, the terms *increasing returns to scale* and *scale economies*, and *decreasing returns to scale* and *scale diseconomies* can be used interchangeably (Banker and Kemerer, 1989).

3. DEA assumes only that a monotonically increasing and convex relationship exists between inputs and outputs. These assumptions ensure that marginal productivity is decreasing so that diminishing returns for small projects would not be followed by increasing returns for large projects.

4. We ran a number of sensitivity checks by removing influential project observations from our data set. Individually, we deleted projects #14, 21, 1, 8, 13, and 16. We also deleted combinations of projects, such as #14 and 21, and #13, 14, and 21. After each deletion, we re-computed the DEA heuristics for the remaining projects. Our results indicated that, for all situations, the sample of remaining projects exhibits increasing returns to scale.