1987

FACTORS AFFECTING SOFTWARE MAINTENANCE PRODUCTIVITY: AN EXPLORATORY STUDYl

Rajiv D. Banker
Carnegie Mellon University

Chris F. Kemerer
Massachusetts Institute of Technology

FACTORS AFFECTING SOFTWARE MAINTENANCE PRODUCTIVITY: AN EXPLORATORY STUDY[1]

Rajiv D. Banker
Srikant M. Datar
Carnegie Mellon University

Chris F. Kemerer
Sloan School of Management
Massachusetts Institute of Technology

ABSTRACT

Systems developers and researchers have long been interested in the factors that affect software development productivity. Identification of factors as either aiding or hindering productivity enables management to take steps to encourage the positive influences and to eliminate the negative ones. This research has explored the possibility of developing an estimable model of software development productivity using a frontier estimation method. The approach taken is based upon output metrics for the entire project life-cycle, and includes project quality metrics. A large number of factors potentially affecting software maintenance productivity were included in this initial investigation. The empirical analysis of a pilot data set indicated that high project quality did not necessarily reduce project productivity. Significant factors in explaining positive variations in productivity included project team capability and good system response (turnaround) time. Factors significantly associated with negative variations in productivity included lack of team application experience and high project staff loading. The use of a new structured analysis and design methodology also resulted in lower short term productivity. These preliminary results have suggested a number of new research directions and have prompted the data-site to begin a full scale data collection effort in order to validate a model of software maintenance productivity.

1. INTRODUCTION

Production of software is generally constrained by what is commonly referred to as the "software bottleneck" -- the high and growing demand for software that has far outstripped the current capacity of software developers. For instance, Zavala (1985) notes that demand for software is growing at 20% to 30% per year while the supply of trained staff is only growing at 3% to 4%. One result of the failure of software supply to keep up with demand is the long "applications backlog" at many large data processing departments. To a large degree, this backlog is caused by the increasing burden that systems maintenance inflicts on data processing departments (Grammas and Klein 1985). According to a wide variety of sources, maintenance programming, the correction and enhancement of existing programs, accounts for anywhere from 50% to 80% of data processing resources (Elshoff 1976; Freedman 1986; Kolodziej 1986). Parikh (1986) estimates that more than $30 billion are spent worldwide on software maintenance. The trend is that the maintenance problem will continue to escalate (Jones 1986). Research into productivity in the maintenance subset of software development[2] has been especially lacking (Lientz 1980).

This research makes an initial contribution to the understanding of software maintenance by providing insights into factors that affect software maintenance productivity. One possible cause of productivity variances is the attention spent in producing a high quality product. A common view is that a tradeoff exists between quality and productivity, that to achieve a high level of either requires sacrificing the other (Case 1985; Kriebel 1979). An alternative hypothesis is that both dimensions are

under the control of the project leader, and that superior project leaders will make productive use of their staffs in ways that do not sacrifice quality (Lambert 1984; Mohanty 1981). An example of this is the use of software tools, such as data dictionaries or code generators, that both relieve project team members of some of the more mundane tasks while improving quality by ensuring consistency. A second research question that we address is the relationship between quality and productivity on a software maintenance project.

This paper reports on our efforts to develop and estimate a preliminary model of the software production process using pilot data from 65 software maintenance projects recently completed by a large regional bank's data processing department. There are four eventual goals of our ongoing research. The first of these goals is to measure factors that affect software *maintenance* productivity, an issue that has not been addressed in the MIS literature. The second goal is to integrate the quality and productivity dimensions of software measurement. Third, in contrast to much other research in this area, the intent is to examine the productivity of entire projects rather than only the programming phase, which typically accounts for less than half the effort on a software project (Boehm 1981). Fourth, we will include variables relating to the quality of labor employed on the projects. Many small studies or controlled experiments have noted the importance of these variables, but relatively few large empirical studies have been able to empirically assess the importance of these factors (Chrysler 1978; Curtis 1981; Sackman 1968).

In order to investigate the set of potential productivity factors, we employ the technique of Data Envelopment Analysis (DEA) to estimate the relationship between the inputs and products of software maintenance. The choice of DEA is motivated by the need to simultaneously consider multiple inputs and products and to not impose an arbitrary parametric form for the underlying production correspondence. Furthermore, DEA estimates the minimum amount of input required, given the size and complexity of the project, rather than the average amount of input which would be estimated using regression-based methods. The former is more meaningful for management control and efficiency evaluation purposes and is consistent with a microeconomic definition of a production function.

The general approach of this research is to model software development as a microeconomic production process utilizing inputs and producing products. This approach is suggested by the work of Kriebel and Raviv (1980; 1982) and Stabell (1982). This general model is best represented by the simple diagram shown in Figure 1.
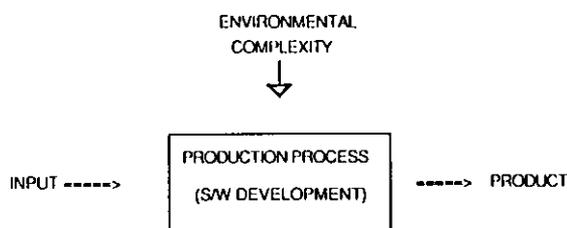

Figure 1. General Production Process Model

The amount of input (e.g., labor hours) required by a software development project depends on the size and complexity of the resulting product and the effects of a number of environmental complexity factors, such as the response time of the development hardware. Since the product is specified in advance to the software project leader, he or she must act to minimize the amount of input required in order to improve productivity.[3] Therefore, the dependent variable, labor hours, is a function of the size and complexity of the product (described in Section 2) and the environmental complexity factors (described in Section 3).

The remainder of this paper has the following structure. The software maintenance inputs and products are described in Section 2. Section 3 compares the environmental variables selected for this research with variables chosen by researchers in new software development productivity. Section 4 describes the model and its estimation. The source of the data and the data collection methods are outlined in Section 5. Section 6 presents the results of the analysis and Section 7 provides some conclusions and suggestions for future research.

2. SOFTWARE DEVELOPMENT INPUTS AND PRODUCTS

The critical input to software development that we focus on is the amount of professional work-hours expended by the project team. Personnel costs constitute at least 45% to 50% of a data processing department's budget (Grammas 1985), and 80% of the department's costs at the current data-site. Since professional data processing staff time is the most expensive and scarce input resource in software development, work-hours has been the variable of interest in most previous studies. Furthermore, the cost of the other major input, hardware, (i.e., CPU time, disk storage, etc.) continues to decline, increasing the ratio of personnel cost to machine cost.

The identification of consistent, quantifiable products from the software development process is probably the single biggest challenge in the field of software metrics. As the final product of any systems development project is a coded program or programs, the traditional measure has been the count of the number of written source lines of code (SLOC).[4] SLOC has the advantage of being easily countable by automated means, in addition to apparently representing the amount of work required to build a system. The SLOC metric, however, is not without its weaknesses. Two common problems are comparing programs written in different languages and comparing the results of studies that have used different counting rules for counting SLOC (Jones 1986). These problems were not an issue in the current research, as all projects were written in COBOL and consistent counting rules were employed.

SLOC, however, is actually the product of only one phase of the project, the programming phase. For new development projects SLOC is generally considered to be an accurate surrogate for all project activities since larger systems typically require both more analysis and more programming than smaller systems. In the case of maintenance projects, this assumption will not, in general, hold. It is easy to imagine a project in a maintenance environment with large amounts of effort expended in analysis and design that result in relatively few additions or changes to lines of code. Therefore, while SLOC is an adequate measure of the size of the coding and testing phase, it is inadequate with respect to the size and complexity of analysis and design on a maintenance project.

We address this problem using Albrecht's Function Point metric (Albrecht and Gaffney 1983). The Function Point metric first counts the number of unique input types, output types, logical files, external interface files, and external queries handled by an application. These counts are then weighted depending upon difficulty and further modified by fourteen "complexity factors" defined by Albrecht.[5] Function Points thus capture the magnitude and complexity of the analysis and design task of various projects.

The use of Function Points as a measure of the product of software development has been validated or suggested by Behrens (1983), Vacca (1985), Jones (1986), Kemerer (1987), Albrecht (1985), Gaffney (1986), and Lambert (1984). In a recent Delphi-type survey by the Quality Assurance Institute (Perry 1986), Function Points per man-month was selected as the leading productivity measurement by a number of Fortune 500 level firms. In summary, the inputs of the general model were implemented with work-hours and the products with Function Points and Source Lines of Code. (See Figure 2.)
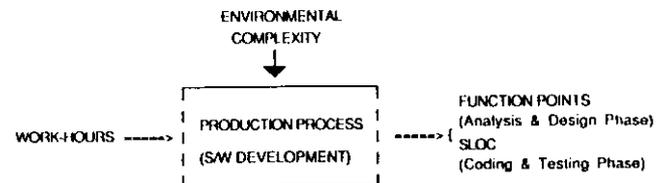
ENVIRONMENTAL
COMPLEXITY

WORK-HOURS -----> | PRODUCTION PROCESS | -----> FUNCTION POINTS (Analysis & Design Phase)
                   | (S/W DEVELOPMENT) | SLOC (Coding & Testing Phase)

Figure 2. Specific Production Process Model

3. ENVIRONMENTAL VARIABLES SELECTION

Previous research on new development has identified a large number of factors which may have an impact on productivity. In addition, detailed discussions with managers at the data-site led to the identification of other factors believed to be important sources of productivity variation. These factors are summarized into the following four categories: personnel, project management, user, and technical environment. A brief discussion of each follows. An additional factor that may influence productivity is the overall quality   of the

162

Table 1 Personal Factor Variables

	a	b	c	d	e	f	g	h	i	j	S	BDK
DP Experien	X	X	X	X	X					X	6	X
Appl Expern	X	X	X			X		X	X		6	X
S/W Expernc				X	X	X		X			4	
H/W Expernc				X		X		X			3	
Facility Expr					X						1	X
Capability								X			1	X
Education					X						1	X
Inhouse %						X					1	X
Parttime %						X					1	
Prog Partic				X							1	
Age					X						1	
Morale										X	1	

a = Gayle (1971); b = Scott (1974); c = Wolverton (1974); d = Walston (1977); e = Chrysler (1978); f = Putnam (1978); g = Albrecht and Gaffney (1983); h = Boehm (1981); i = Rubin; j = Jones (1986); S = Summary; BDK = Banker, Datar and Kemerer.

product produced. This variable has generally not been included in previous empirical studies of productivity and is discussed separately in Section 3.5.

3.1 Personnel Variables

Personnel variables are widely believed to be critical in affecting the productivity performance of a project team. Table 1 is the first of four tables showing the selection of productivity variables by other researchers. Each row represents a variable, and each column a researcher. An "X" indicates that the variable was used by the researcher. A summary column shows the number of previous researchers using a particular variable, and the last column ("BDK") designates whether it was used in the current research. From Table 1, it is apparent that the experience of project team members, measured along one or more dimensions, is believed to be a critical element. For this study, each project team member's total data processing experience, his data processing experience at this facility, and his experience with each application were recorded. As all of the projects were written

in COBOL for IBM mainframes, the facility experience data obviated the need to collect software and hardware experience.

Capability, or some measure of talent, is often discussed but rarely used in research of this type due to difficulties in measurement. In this research, staff capability was captured through the use of the personnel review system at the data-site. Each staff member is given a yearly review that is summarized in a numerical score ranging from 1 (best) to 5 (worst). These data are used as a measure of capability or skill.

We also collected information on the highest level of education and the amount of in-house versus outside contractor staffing. However, the hiring and personnel policies at the data-site generated a very homogeneous dataset, and therefore education and in-house percentage were dropped as potential variables. The remaining variables were each used by only one of the ten previous researchers, and were not felt to be either important or measurable at the data-site. Therefore, the variables included in the model were capability, application experience, and data processing experience.

Table 2. Project Management Factor Variables

	a	b	c	d	e	f	g	h	i	j	S	BDK
Schd Constr						X	X				2	X
Staff Load				X		X					2	X
Travel									X	X	2	
Communictn		X									2	

a = Gayle (1971); b = Scott (1974); c = Wolverton (1974); d = Walston (1977); e = Chrysler (1978); f = Putnam (1978); g = Albrecht and Gaffney (1983); h = Boehm (1981); i = Rubin; j = Jones (1986); S = Summary; BDK = Banker, Datar and Kemerer.

3.2 Project Management Variables

Project management variables, including schedule constraints, staff loading, travel requirements, and project communication are less well represented in the literature. (See Table 2.)

Schedule variables are among the most critical that may be under a project manager's direct control. This research also recorded the calendar duration of the project in order that the loading, or work-months per calendar month, could be calculated. None of the projects in the dataset required any travel, and their small size in terms of the number of staff (average project size for the 65 projects was 2.6 people) meant that intra-project communication was not a critical issue. Therefore, the variables considered were deadline pressure and manpower loading.

3.3 User Variables

Although Lientz and Swanson (1981) have discussed the potential importance of user variables, Table 3 shows that user variables have played only a limited role in previous empirical studies.

Table 3. User Factor Variables

	a	b	c	d	e	f	g	h	i	j	S	BDK
High Reliab								X	X	X	3	X
Reqmt Volat	X		X						X		3	X
User Partic				X					X		2	X
# User Orgn									X		1	X
Usr DP Knwl									X		1	X
Usr Appl Kn				X							1	X

a = Gayle (1971); b = Scott (1974); c = Wolverton (1974); d = Walston (1977); e = Chrysler (1978); f = Putnam (1978); g = Albrecht and Gaffney (1983); h = Boehm (1981); i = Rubin; j = Jones (1986); S = Summary; BDK = Banker, Datar and Kemerer.

Table 4. Technical Environment Factor Variables

	a	b	c	d	e	f	g	h	i	j	S	BDK
Mod Prog Pr		X	X			X		X		X	5	X
Tools		X				X		X		X	4	X
Respons Time			X			X		X	·		4	X
Language		X				X	X			X	3	X
Volatility			X					X			2	X
Reusbl Code										X	1	
Classified			X								1	
Distance	X										1	
Exist Docum											0	X

a = Gayle (1971); b = Scott (1974); c = Wolverton (1974); d = Walston (1977); e = Chrysler (1978); f = Putnam (1978); g = Albrecht and Gaffney (1983); h = Boehm (1981); i = Rubin; j = Jones (1986); S = Summary; BDK = Banker, Datar and Kemerer.

Earlier research suggested two important user variables -- high user-required reliability (the importance placed on avoiding system failure) and requirements volatility (the degree to which the user-stated requirements changed over the course of the project). Additionally, discussion with staff members at the data-site indicated the perceived importance of the following variables: the degree of user participation in the project, number of user organizations having signoff responsibility, the user's data processing knowledge, and the user's application knowledge. Information on these variables was obtained from the project leader and validated by his or her section head. Controls on this data collection are described in Section 5.

3.4 Technical Environment Variables

Technical environment variables, shown in Table 4, have a long history of inclusion in productivity models. This likely reflects practitioners' hopes for technological solutions to the productivity problem and researchers' attempts to find those solutions. Five variables of interest suggested by the literature are the use of modern programming practices, use of software tools, response time, choice of language and hardware/ software volatility. Hardware/software volatility reflects the amount of change in the underlying environment in which the application is being written.

Use of reusable code was in its infancy at the data-site during the period when data were being collected, and insufficient data were available on its use. None of the work at the data-site was classified, and distance to the machine room has ceased to be a variable of interest with modern teleprocessing. We included one new variable, good quality documentation, that is perceived to be significant in a maintenance environment. The quality of the documentation was rated by the project leader. Of the variables measured, tools and language were dropped due to lack of variance in the data. Therefore, four technical environment variables were included in the model: use of modern structured analysis, design, and programming practices; presence of good interactive response or batch turnaround time; hardware or software volatility, and good documentation.

3.5 Measurement of Project Quality

While the emphasis on measuring the size of a system is clearly critical to productivity measurement, it could be argued that a size metric, such as SLOC, without a measure of the quality of those

165

lines, is insufficient. Two important dimensions of program quality are adherence to specifications and freedom from defects.

A significant body of literature exists on the construct of user satisfaction. Unfortunately, the focus of that research is general user satisfaction with a data processing department, whereas our focus is on user satisfaction across projects within the same department. However, a survey instrument for measuring user project satisfaction has been developed by Powers (1971) and was later validated by McKeen (1983). This instrument was used in the current research.

The converse problem exists in the software quality research literature; that is, many of the metrics developed in this area have generally been too specific, at the level of a line of code or groups of lines of code within a program.[6] The data required for this micro level of detail were not available at the data-site. However, a recent survey (Perry 1986) by the Quality Assurance Institute suggests three quality metrics as the most widely accepted in industry. These are user perceived functional quality, user software satisfaction, and production jobs processed without incidence. The first two of these are covered by the Powers instrument. The third idea was developed into a site-specific quality metric that rated projects as average, above average, or below average with respect to the problems encountered after the project's software was turned operational. This metric is described in Section 5.2.3.

4. EXPLORATORY MODEL AND ESTIMATION

The primary purpose of this initial analysis is to investigate the potential impact of a number of potential productivity factors. We model the actual input resources (labor hours) used as a multiplicative function of the primary production correspondence and the environmental factors. This general model is similar to others developed in the literature (Albrecht and Gaffney 1983; Boehm 1981). It should be noted that since the product requirements are prespecified, we model the primary production correspondence as the minimum amount of input resources required to produce the prespecified product, which is described in terms of Function Points and SLOC. Since our objective is to estimate the minimum (rather than the average) consumption of input resources, we adopt an extremal or frontier estimation technique, Data

Envelopment Analysis (DEA). DEA uses a linear programming approach to identify the most efficient projects (Banker, Charnes and Cooper 1984; Charnes, Cooper and Rhodes 1981).

DEA is an appropriate tool for this purpose for several reasons. First, since no preset standards exist, productivity needs to be evaluated relative to other projects, which is the basis of the DEA efficiency rating. Second, software development produces multiple products, so that simple partial productivity ratio measures are insufficient. Third, DEA does not impose a parametric form on the production function and only assumes a monotonic and convex relationship between inputs and products. Given the limited knowledge about the production process underlying software development, specifying a parametric form such as Cobb-Douglas (Stabell 1982) for the production correspondence is difficult to substantiate theoretically or validate statistically and it is not immediately apparent what restrictions these hypotheses, treated as axioms in the econometric approach, impose on the production correspondence.

We next explore the average impact of different environmental factors on the DEA efficiency rating. Since our objective is to identify the average impact of environmental factors on productivity, we use multivariate regression analysis. The general idea is that two projects could be identical in terms of their outputs, yet one may have environmental factors (such as poor hardware response time) that causes that project to consume more labor hours. The latter project will be rated as inefficient relative to the former, since both produced the same outputs but the second required more inputs. The purpose of the analysis will be to isolate and measure the factors that may have influenced the productivity ratings.

5. DATA COLLECTION

5.1 Data Source

Data for this research were collected at a large regional bank's data processing department. The types of applications represented are typical financial transaction processing systems, and are written in COBOL to run on IBM hardware. COBOL and IBM are the most widely used software and hardware in commercial data processing and therefore this site is likely to be representative of much of current business data processing.[7] The

data processing department is divided into eighteen "sections," which are organized around common sets of applications. Three of the sections were selected by the Bank as representative of the department as a whole. Two criteria were used to select projects completed by these three sections: size and recency. Selecting larger projects allows the examination of the projects that consume the bulk of the Bank's resources. Project size is also important in that the factors affecting productivity on short, one person projects are likely to be overwhelmed by individual skill differences across project staff members (DeMarco 1982; Sackman, Erikson and Grant 1968). We only considered "significant" projects at the Bank that cost a minimum of $5,000 in internal dollars.

Project recency is important for two reasons. Since data were collected retrospectively, old projects were not included because personnel turnover and lack of documentation retention made data collection impossible. Second, using only recent projects legitimizes cross project comparisons in that the technology and personnel involved are likely to be very similar. After discussions with Bank staff, only projects completed within the 18 month period between January 1, 1985 and July 1, 1986 were included in the study. Data were collected during the summer of 1986. Due to a number of factors, including reorganizations, the conversion to a new time reporting system, use of contractors, personnel turnover, and the elimination of a few unsuitable (i.e., non-COBOL) projects, complete data were available for only 65 of the 84 potential projects. These 65 projects have the characteristics shown in Table 5.

5.2 Data Collection Methods and Controls

This section describes the main data types and how they were collected. When possible, we attempted to use data already collected and employed by the Bank, rather than developing new data collection instruments that would impose additional burden on Bank staff.

5.2.1 Professional work-hours

The key input variable was the number of work-hours charged by project by person. Previous research has generally been satisfied with work-hours by project only. The limitation of that approach is immediately apparent if a 1000 work-hour project staffed by a team of veteran programmer/analysts who were also intimately familiar with the application being provided is compared with one staffed by a team of novices. Intuition suggests that the former team is likely to be more productive, yet much prior research has treated both of these simply as "two 1000 work-hour" projects. This paper characterizes the actual work-hours expended along a number of dimensions, particularly experience and capability.

The characterization of work-hour data was accomplished via a personnel survey that requested each project member to fill in data on his or her total data processing experience, data processing experience at the Bank, application experience, and education. These forms were matched to the records in time reporting via an employee number. Forms were not received from all project members charging time, chiefly due to the fact that the

Table 5. Project Summary Data

	MEAN	STAN. DEV.	MINIMUM	MAXIMUM
Work-hours	937	717	130	3,342
Func Points	118	126	8	616
SLOC	5,415	7,230	50	31,060
Duration	6	4	1	22

individual had transferred or left the Bank or had been an outside contractor. In the event that the hours on a project could not be categorized, that project was dropped from the study.

5.2.2 Product size and environmental complexity factors

Product size and environmental complexity data were collected via a survey of project leaders. The size data collection form captured data on

o Function Points
o New and modified source lines of code

while the environmental complexity form captured data on

o Function Point complexity
o Project management
o User factors
o Technical environment

Due to the broad nature of the phenomenon modeled, a large number of factors were identified as possible variables. In order to make the data collection effort feasible at the field site, most of the factors were measured in only one way. This raises the question of whether any factors were not shown to be significant due to method variance. One control that was used to mitigate this was to use questions drawn from previous research whenever possible.

A number of steps were taken to assure that the data collection forms would be filled out as accurately as possible. A training session to walk through the data collection forms was held for all project leaders and their section heads. In addition, a member of the research team was on-site during the entire data collection process and provided ad hoc support to project leaders. An automated tool was also available to aid in the counting of source lines of code.

The following controls were established to attempt to provide additional assurances of data validity. After the project leader had completed the data collection form, it was first reviewed by the section head. As each project was compared only to other projects within the same section, the review by the section head also ensured consistency across projects. After review by the section head, the data collection form was forwarded to the research

team for a second review for completeness and reasonableness.

5.2.3 Quality data collection

An important issue in measuring productivity is whether the products of efficient projects are of the same quality as those of less efficient projects. This study addressed two questions as adjuncts to the efficiency measure generated. These metrics should not be confused with general measures of systems effectiveness.

The first quality concept is that of operational quality, whether the system operates smoothly once it is implemented. This measure was generated by a staff section within the Bank from three existing sources:

o daily abnormal end (ABEND) report
o weekly section status reports
o ad hoc user problem reports

Data from the two month period following implementation were compared with data from the previous twelve months' trend. Significant deviations resulted in above or below average operational quality ratings. The control for this measure was to forward the ratings for each section to the appropriate section head for review.

The second quality concept is that of user project satisfaction. A survey, based on the Powers instrument, was sent to the users who had requested the individual projects. These forms were returned directly to the research team without review by the sections.

6. DATA ANALYSIS

There were two broad objectives to the data analysis: 1) to determine the appropriateness of the general approach of using DEA for software development analysis, and 2) to identify which factors in our pilot data sample seemed to be the most important and therefore merit further investigation in future research. The results of these analyses are presented below.

6.1 DEA Efficiency Ratings

A DEA efficiency score for each project was developed using metrics for total work hours, Function Points, and SLOC. Three separate primary

Table 6. Summary of DEA Results

	MAXIMUM	MINIMUM	MEAN	ST. DEV.	n
Section M	1.00	.19	.62	.28	19
Section D	1.00	.14	.62	.30	27
Section L	1.00	.14	.62	.32	19

production functions were estimated using DEA (one for each section). This was done to ensure that projects were being rated only against similar applications.

One interesting result was that all three sections showed wide variations in productivity, as shown in Table 6.

This is consistent with much of the literature on software development productivity, particularly that dealing with individual differences. In addition, the distribution of efficiency results within each section is consistent across sections, which supports the pooling of the individual section results in the multivariate regression analysis.

We also estimated a linear regression model (consistent with that of other researchers) with work-hours as the dependent variable and Function Points and total SLOC (new SLOC plus modified SLOC) as the independent variables. This model (presented below) showed that these two measures of size were excellent predictors of total effort.

Actual work-hours = 355.0 + 3.49(FP) + .03(SLOC)
                          (.11)   (7.25)   (3.76)

$R^2 = 69.9\%$ $(\overline{R}^2 = 68.9)$

One concern with this model might be multicollinearity, as previous research on new development projects has shown a correlation of .94 and greater between Function Points and SLOC (Albrecht and Gaffney 1983). Our earlier discussion suggests that Function Points and SLOC are not likely to be as highly correlated in the case of maintenance projects. Indeed, the correlation between Function Points and SLOC is .57 in this dataset.

An alternative model was also estimated, utilizing three independent variables: New SLOC, Modified

SLOC, and Functions Points. This three variable model did not exhibit appreciably more explanatory power than the two variable model (R^2 of .698 versus .689). The null hypothesis of equality of the estimated coefficients for New SLOC and Modified SLOC could not be rejected at the 10% level (Pindyck and Rubinfeld 1981). Therefore, the three variable model was not pursued in the interests of parsimony. In summary, we conclude that the original assumptions regarding the choice of metrics sufficiently represent the product of the software maintenance process. We use the DEA efficiency ratings to examine the effects of the environmental variables on productivity.

6.2 Multivariate Regression Results

The reciprocal of the DEA efficiency score is regressed against the environmental variables described in Section 3 in a multivariate regression model. A summary of this model appears in Table 7, and we discuss each of the significant variables in turn.

The dependent variable is the reciprocal of the DEA efficiency score. Therefore, the interpretation of the signs of the coefficients is that positive (+) signs show reduced productivity, while negative (-) signs show increased productivity. The R^2 for the entire sixteen variable model is .53 (F-value of 3.32 is significant at the 1% level). The value of the intercept was 1.9 (t=1.98). The Belsley-Kuh-Welsch (1980) test did not indicate any multicollinearity problems.

It should be noted that the R^2 indicates the amount of variation in productivity explained. Other researchers have explained the variation in hours, where the independent variables have included size. For our dataset, an analogous regression model with hours as the dependent variable and size and environmental complexity as the independent variables produces an $R^2 = .85$ (F = 14.02).

169

Table 7. Summary of the Regression Model

VARIABLE	Beta	t	p	DESCRIPTION
TOPSTAFF	-.01	-2.43	.02	% Hours charged by best staff
LOAPPEXP	+.80	+2.08	.04	High % of application novices
LODPEXP	+.65	+1.42	.16	High % of dp novices
LOADING	+.42	+2.37	.02	Work-months/Calendar Months
TIGHTDEAD	-.79	-2.25	.03	> average deadline pressure
INTERNAL	-.79	-1.57	.12	No external user
LOUSERDP	-.69	-1.32	.19	< average user dp knowledge
STAFFAPT	+.61	+1.18	.24	Poor communications with user
LOUSERAP	+.55	+1.14	.26	< average user appl knowledge
HIGHRELY	+.34	+0.91	.37	High required reliability
LOAGREE	+.28	+0.62	.54	Low initial user agreement
GOODRESP	-.86	-2.28	.03	Good response/turnaround time
STRCMETH	+.83	+2.09	.04	New structured method used
VOLATLTY	+.52	+1.49	.14	Many s/w environment changes
GOODDOC	+.09	+0.24	.81	Good documentation available
QUALITY	+.06	+0.18	.86	3 point scale hi=above average

6.2.1 Personnel variables

The personnel variables are often cited as being critical to software productivity, and these results suggest that they are important in software mainte- nance. The most significant variable, however, is one that is often discussed but rarely measured: the capability of the project team, here listed as TOPSTAFF. The Bank gives a yearly review to all staff members and the review is summarized in the form of a numerical score, from one to five, one being the best. The variable TOPSTAFF is the percentage of hours charged to projects by individuals rated one or two. The higher percentage of these high capability staff members, the better the productivity.

Staff members were classified as application novices if they had less than 24 months experience on an application prior to the project. Previous research by Jeffery and Lawrence (1985) has shown that additional experience beyond 24 months does not seem to result in increased productivity. Project staff at the Bank had indicated that a minimum of one non-novice individual was necessary in order to "leverage" his or her skills over any application novice team member. The absence of this desirable situation was quantified by setting a dummy variable, LOAPPEXP, equal to one when 90% or more of the hours were charged by application novices. This variable was significant at the 5% level. The significance of project team capability and application experience in explaining software maintenance productivity is consistent with much prior research on new software development.

Similarly, staff members were categorized as data processing novices if they had less than 24 months of data processing experience. The experience could include non-Bank data processing experience, although, due to the Bank's hiring policies, the vast majority of staff members in this study had only Bank experience. Projects with a majority of data processing novices were expected to be less productive than those not so burdened. According- ly, the dummy variable LODPEXP equals one when 50% or greater of the hours charged were by data processing novices. LODPEXP was not as significant as LOAPPEXP[8] in explaining variations in software maintenance productivity.

6.2.2 Project management variables

The first significant project management variable in this analysis is LOADING, the rate at which people are added to the project. LOADING is equal to the

total number of work-months divided by the total project duration in calendar months. Higher loading indicates a greater amount of parallelism on the project, plus a possible increase in the amount of project communications. This was found to have a negative impact on productivity at the 5% level, and is consistent with the results of researchers on new software development.

A second significant variable is deadline pressure. Project leaders and their section heads were asked on their surveys whether there was greater than average deadline pressure on the project. This variable, TIGHTDEAD, was found to be a significant boost to productivity, at least in the short run sense indicated by this measure. The explanation seems to be that increased deadline pressure reduces, at least for the duration of the project, some amount of the slack that is present in any organization. Whether an organization would want to pursue this tactic as a long-term strategy is questionable, however, given the likely deleterious effect on morale and the resulting increase in turnover. This is particularly important in light of the significance of the application experience variable.

6.2.3 User variables

In general, the significance of the user variables was low.[9] Given the Bank staff's a priori suggestions, this was a surprising result. It may be that the impression that poor user relationships leave with project leaders is greater than their actual effect on project productivity.

The most significant of the user variables was INTERNAL. A survey question concerning the number of user signoffs required was designed to identify those projects that needed to reach agreement across multiple users. In terms of responses, however, very few projects had greater than one user, while a significant number of projects were internally generated, typically to increase efficiency or throughput on an application. This variable was coded as a zero-one dummy, where INTERNAL = 1 meant that no outside users were involved. The fact that these projects may be more efficient is not surprising, since the removal of the need to communicate specifications across departments and the likely reduced documentation burden would aid in increasing efficiency of product development.

6.2.4 Technical environment variables

The most significant variable in the list of Technical Environment variables was GOODRESP, a dummy variable indicating either an interactive development environment or good (< 4 hours) batch turnaround. This had the effect of improving productivity, which is intuitive, and consistent with some limited research in this area (Boehm 1981; Lambert 1984).

A second significant variable was the dummy variable STRCMETH, which indicated the use of a structured analysis and design methodology based on the Gane/Sarson principles and tools. Projects using this methodology were less productive than those that did not, an initially eye-opening result for the managers at the Bank, but one that actually makes a good deal of sense upon close scrutiny. What is being measured is a snapshot of short-term productivity, not long-term productivity. Many of the benefits of using a detailed methodology that requires a lot of documentation are not observed until the next project, when enhancement or repairs need to be made to the system. In the short term, the extra effort is not necessarily going to show any benefit, and the extra hours will show up as reduced productivity. Additionally, it should be added that use of this methodology was new at the Bank, and was, at least for one of the sections, exactly coincident with the projects collected in this dataset. Therefore, this factor may also exhibit a learning curve.

VOLATLTY, defined as frequent changes to the hardware/software environment, either every few weeks for major changes or every few days for minor changes, was only marginally significant. This variable was also shown to reduce productivity (consistent with other researchers, see Boehm 1981), although only at the 15% significance level.

Good documentation (GOODDOC) had been suggested by managers at the Bank as a potential important factor in explaining productivity. As shown in Table 7, it was not a significant factor.

6.2.5 Quality as a productivity variable

One remaining question about these productivity measures is the relationship between the most productive projects and quality: Do projects with high quality exhibit high productivity, or is high productivity attained only by sacrificing quality?

171

Table 8. Operational Quality Versus Productivity

	Low Productivity	Med Productivity	High Productivity
Low Quality	3	1	5
Med Quality	15	12	16
High Quality	3	8	2

Table 9. User Project Satisfaction Versus Productivity

	Low Productivity	Med Productivity	High Productivity
Above Average Satisfaction?			
No	7	3	8
Yes	6	12	9

A first attempt to answer this question would involve adding the two quality metrics, operational quality (QUALITY) and user project satisfaction (CUSTACCP), as independent variables in the multiple regression analysis. Unfortunately, only 45 of the 65 user surveys were returned, and therefore CUSTACCP was not employed as an independent variable. However, QUALITY was added and was not significant. (See Table 7.) A second approach was to cross-tabulate the data as shown in Tables 8 and 9.

Table 8 shows the operational quality data versus an aggregation of the productivity data. For low quality projects, there is an approximately equal chance of a low or high productivity rating, and similarly for high quality projects. This explains why the quality variable is not found to be significant in the exploratory multivariate regression analysis. Therefore, the data from this data-site do not support the hypothesis that achieving high productivity or quality requires sacrificing the other.[10]

Table 9 was constructed by converting a five point scale on which data for user project satisfaction were converted into a dummy variable, above average user project satisfaction. As with operational quality, a relatively random spread of quality-productivity occurrences is seen.[11]

7. CONCLUSIONS AND FUTURE RESEARCH

This paper explored the potential of developing a DEA-based model of software maintenance productivity and sought to identify productivity factors that merit further study. The estimation of this model using pilot data collected from a large commercial bank have suggested several interesting insights. Our analysis indicates that the factors that affect new software development (particularly personnel experience and capability) also seem to play an important role in software maintenance. Our analysis suggests that high productivity appears to be possible in a maintenance environment without sacrificing quality, and that the quality/productivity relationship bears further investigation.

This research has raised many questions which suggest possible avenues for future research. An interesting methodological extension would be the simultaneous consideration of output and input variables as well as the environmental factors in a single model. Another methodological extension would involve estimating a stochastic frontier using techniques of Stochastic DEA. This involves a composed error formulation with a two-sided random component and a one-sided error caused by inefficiencies.

Another area for further work stems from the fact that the productivity measures used in this analysis are clearly short-term. The long-term impact on productivity of some of these factors (particularly the use of structured analysis and design methodologies) would be an interesting extension. The notion of long-term productivity is related to quality in that a better quality product today should result in less maintenance in the future. Research could be directed at modeling software quality as a primary goal, rather than as an adjunct as was done here. Finally, a larger and richer dataset could allow more detailed examination of the factors and their possible interplay, an exercise not really feasible with the limited amount of data available in this study.

ENDNOTES

[1] This research was funded in part by the Center for the Management of Technology and Information in Organizations, Graduate School of Industrial Administration, Carnegie-Mellon University, and the International Business Machines Corporation.

Helpful comments from four anonymous referees are gratefully acknowledged.

[2] The term "development" is used here in its most general sense, which includes maintenance programming. The term "new development" will be used in this paper to describe programming that is strictly the generation of new code.

[3] Note that this is in contrast to many other production settings where the manager has fixed inputs and desires to maximize output.

[4] Computer scientists have also developed what might be termed "micro" software metrics, those below the level of a source line of code. Examples of these would be the software science metrics of Halstead (1977) and the complexity measure of McCabe (1976). These metrics have generally not been applied to large scale software productivity due to difficulties in measurement.

[5] These are data communications, distributed processing, application performance objectives, heavily used configuration, high transaction rate, online data entry, end user efficiency, online update, complex processing, reusability, installation ease, operational ease, multiple sites, and flexibility (see Albrecht 1984).

[6] See, for example, the survey by Mohanty (1979).

[7] Of course, while the dataset contains 65 projects, they were all gathered within one organization. Therefore, the external validity of the results remains to be demonstrated.

[8] The correlation coefficient between LODPEXP and LOAPPEXP is .30.

[9] We tested the possibility of high correlation among the user variables. The highest correlations between user variables (and the highest correlation of any independent variables) were between STAFFAPT and LOAGREE (.46) and between STAFFAPT and INTERNAL (.43). Six separate regressions were run with one user variable as the dependent variable and the other five as independents, varying the dependent variable on each run. The best fit (R^2 =.36) was for the STAFFAPT variable. While this degree of correlation was not believed to be large, a run of the main productivity model was made, omitting STAFFAPT. The t-statistic for LOAGREE improved, but not enough to make it a significant variable at the 10% level.

[10] The chi-squared test value is 7.93, significant at the 10% level. The explanation for this relatively high value is that there seems to be some drift towards average productivity (neither high nor low) when the quality is high. Note that the chi-square test may be inappropriate in this case, given that 2/3 of the cells have expected values less than five.

[11] The chi-squared test value is 3.89, significant at the 15% level. The explanation of these results given in footnote 10 applies here as well, mutatis mutandis.

REFERENCES

Albrecht, A. J. *AD/M Productivity Measurement and Estimate Validation.* CIS & A Guideline 313, IBM Corporate Information Systems and Administration, November 1, 1984.

Albrecht, A. J. "Function Points Help Managers Assess Application, Maintenance Values." *Computerworld Special Report on Software Productivity,* CW Communications, 1985, pp. SR20-SR21.

Albrecht, A. J., and Gaffney, J. 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.

Banker, R.; Charnes A.; and Cooper, W. "Some Models for Estimating Technical and Scale Inefficiencies in DEA." *Management Science*, Vol. 30, No. 9, September 1984, pp. 1078-1092.

Behrens, C. A. "Measuring the Productivity of Computer Systems Development Activities with Function Points." *IEEE Transactions on Software Engineering*, SE-9, No. 6, November 1983, pp. 648-652.

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

Boehm, B. W. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

Case, A. F. "Computer-Aided Software Engineering." *Database*, Vol. 17, No. 1, Fall 1985, pp. 35-43.

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*, Vol. 27, No. 6, June 1981, pp. 668-697.

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

Curtis, B. "Substantiating Programmer Variability." *Proceedings of the IEEE Conference*, Vol. 69, No. 7, July 1981, p. 846.

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

Elshoff, J. L. "An Analysis of Some Commercial PL/1 Programs." *IEEE Transactions on Software Engineering*, Vol. SE2, No. 2, 1976, pp. 113-120.

Freedman, D. H. "Programming Without Tears." *High Technology*, Vol. 6, No. 4, April 1986, pp. 38-45.

Gaffney, J. E. "The Impact on Software Development Costs of Using HOL's." *IEEE Transactions on*
Software Engineering, Vol. SE-12, No. 3, March 1986, pp. 496-499.

Gayle, J. B. "Multiple Regression Techniques for Estimating Computer Programming Costs." *Journal of Systems Management*, Vol. 22, No. 2, February 1971, pp. 13-16.

Grammas, G. W., and Klein, J. R. "Software Productivity as a Strategic Variable." *Interfaces*, Vol. 15, No. 3, May-June 1985, pp. 116-126.

Halstead, M. H. *Elements of Software Science*. Elsevier, New York, 1977.

Jeffery, D. R., and Lawrence, M. J. "Managing Programming Productivity." *Journal of Systems and Software*, Vol. 5, 1985, pp. 49-58.

Jones, C. *Programming Productivity*. McGraw-Hill Book Company, New York, 1986.

Kemerer, C. F. "An Empirical Validation of Software Cost Estimation Models." *Communications of the ACM*, Vol. 30, No. 5, May 1987, pp. 416-429.

Kolodziej, S. "Gaining Control of Maintenance." *Computerworld Focus*, Vol. 20, No. 7A, February 19, 1986, pp. 31-36.

Kriebel, C. H. "Evaluating the Quality of Information Systems." In N. Szyperski and E. Grochla (eds.), *Design and Implementation of Computer Based Information Systems*, Sitjhoff & Noordhoff, The Netherlands, 1979, Chapter 2, pp. 29-43.

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

Kriebel, C. H., and Raviv, A. "Application of a Productivity Model for Computer Systems." *Decision Sciences*, Vol. 13, April 1982, pp. 266-284.

Lambert, G. N. "A Comparative Study of System Response Time on Program Developer Productivity." *IBM Systems Journal*, Vol. 23, No. 1, 1984, pp. 36-43.

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

Lientz, B. P., and Swanson, E. B. "Problems in Application Software Maintenance." *Communications of the ACM*, Vol. 24, No. 11, November 1981, pp. 763-769.

McCabe, T. "A Complexity Measure." *IEEE Transactions on Software Engineering*, Vol. SE-2, December 1976, pp. 308-320.

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

Mohanty, S. N. "Models and Measurement for Quality Assessment of Software." *ACM Computing Surveys*, Vol. 11, 1979, pp. 251-275.

Mohanty, S. "Software Cost Estimation: Present and Future." *Software: Practice and Experience*, Vol. 11, 1981, pp. 103-121.

Parikh, G. "Restructuring Your COBOL Programs." *Computerworld Focus*, Vol. 20, No. 7A, February 19, 1986, pp. 39-42.

Perry, W. E. *The Best Measures for Measuring Data Processing Quality and Productivity*. Quality Assurance Institute, 1986.

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

Powers, R. F. *An Empirical Investigation of Selected Hypotheses Related to the Success of Management Information System Projects*. Unpublished Ph.D. thesis, University of Minnesota, April 1971.

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.

Rubin, H. A. *Using ESTIMACS E*. Management and Computer Services, Inc., Valley Forge, PA.

Sackman, H.; Erikson, W. J.; and Grant, E. E. "Exploratory Experimental Studies Comparing Online and Offline Programming Performance." *Communications of the ACM*, Vol. 11, No. 1, January 1968, pp. 3-11.

Scott, R. F., and Simmons, D. "Programmer Productivity and the Delphi Technique." *Datamation*, Vol. 20, No. 5, May 1974, pp. 71-73.

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

Vacca, J. "Function Points: The New Measure of Software." *Computerworld XIX*, No. 46, November 18, 1985, pp. 99-108.

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

Wolverton, W. R. "Cost of Developing Large Scale Software." *IEEE Transactions on Computers*, Vol. 23, June 1974, pp. 615-634.

Zavala, A. *Research on Factors that Influence the Productivity of Software Development Workers*. Final Report 4677-85-FR-68, SRI International, June, 1985.