MANAGEMENT SCIENCE
Vol 37, No. 1, January 1991
Printed in U S A

A MODEL TO EVALUATE VARIABLES IMPACTING THE PRODUCTIVITY OF SOFTWARE MAINTENANCE PROJECTS*

RAJIV D. BANKER, SRIKANT M. DATAR AND CHRIS F. KEMERER
Carlson School of Management, University of Minnesota, Minneapolis, Minnesota 55455
Graduate School of Business, Stanford University, Stanford, California 94305 *and*
School of Business Administration, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213
Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

The cost of maintaining application software has been rapidly escalating, and is currently estimated to comprise from 50–80% of corporate information systems department budgets. In this research we develop an estimable production frontier model of software maintenance, using a new methodology that allows the simultaneous estimation of both the production frontier and the effects of several productivity factors. Our model allows deviations on both sides of the estimated frontier to reflect the impact of both production inefficiencies and random effects such as measurement errors.

The model is then estimated using an empirical dataset of 65 software maintenance projects from a large commercial bank. The insights obtained from the estimation results are found to be quite consistent for reasonable variations in the specification of the model. Estimates of the marginal impacts of all of the included productivity factors are obtained to aid managers in improving productivity in software maintenance.

(SOFTWARE MAINTENANCE; SYSTEMS DEVELOPMENT LIFECYCLE; PRODUCTIVITY; DATA ENVELOPMENT ANALYSIS)

1. Introduction

While advances in technology have reduced the per unit cost of computing power, information systems expenditures have been rapidly rising. One of the major reasons for this increase has been the escalating cost of maintaining application software, an expense that currently is estimated to comprise from 50–80% of corporate information systems department budgets (Elshoff 1976, Kolodziej 1986, Freedman 1986, Gallant 1986). In this context maintenance refers to the correction, perfection and adaptation of currently existing software, and therefore includes what are commonly referred to as enhancements (Lientz and Swanson 1980).

In spite of the large and increasing expenditures on software maintenance in industry there exists a relative dearth of academic research on the subject, according to a number of researchers (Lientz and Swanson 1980, Vessey and Weber 1983).[1] Lientz and Swanson, in particular, have called for more software maintenance research, particularly in the area of the "measurement and monitoring of the maintenance process" (1980, p. 162).

Given the large sums spent on software maintenance and the current inability of practitioners to keep up with the growing applications backlog, it seems particularly appropriate to concentrate on the *productivity* of software maintainers. From management's perspective what would be especially useful would be the identification of factors under managerial control that have a significant positive or negative impact on

* Accepted by George P. Huber; received January 1989. This paper has been with the authors 4 months for 2 revisions.

[1] Other research in the modeling of software maintenance has typically focused on fault prediction (e.g., Mellor 1983, Adams 1984) or in the behavior of a single system over time (e.g., Belady and Lehman 1976). Our research is a cross-sectional study of the factors impacting the productivity of significant software maintenance projects.

0025-1909/91/3701/0001$01.25

productivity (Dickson, et al. 1984). Once identified, management can take steps to retain and amplify the positive factors and eliminate or at least reduce the negative factors.

Software maintenance, like many other complex activities, may be difficult to understand, and related problems difficult to diagnose without the aid of a model. In this research we develop an estimable model of software development[2] viewed as an economic production function. This view focuses on the inputs and outputs of the software development process. Unlike previous related software research, we focus on the maintenance of existing software and explicitly incorporate the multi-dimensionality of systems development outputs. Previous research in this area, perhaps due to the unavailability of multi-dimensional empirical data, has tended to adopt single, surrogate measures for the entire process in order to accommodate more tractable models.

In our earlier exploratory study (Banker, Datar and Kemerer 1987) we estimated the values of the coefficients of a two stage model. In the first stage we adopted a production frontier approach to relate labor hours incurred on various projects to the size and complexity of projects using Data Envelopment Analysis (DEA). The one-sided deviations from observed best practice provide measures of productivity which are then regressed against various factors (such as personnel and tool usage) affecting productivity to explain productivity variations.

In the current paper we employ the recently developed Stochastic Data Envelopment Analysis (SDEA) methodology (Banker 1990) which, in addition to productivity related deviations, also allows for the impact of random effects such as measurement errors that may cause deviations on either side of the production frontier.

Factors affecting maintenance productivity directly influence the amount of labor input required for any given level of outputs produced suggesting that in developing the production model we should simultaneously consider the relationship between inputs, outputs and other factors. We accomplish this by extending the SDEA methodology, which also permits us to evaluate the marginal impact of each of the factors influencing maintenance productivity. We also compare the results using this extended SDEA model with more conventional composed error econometric models of Aigner, Lovell and Schmidt (1977) and Meeusen and van der Broeck (1977) to examine the robustness of our results to the specification assumptions.

After presenting our general model for evaluating the factors affecting software maintenance productivity, we then estimate the values of the coefficients using an empirically collected dataset of sixty-five software maintenance projects from a large commercial bank. In this analysis we examine the effects of five factors in the areas of project team member ability and application experience, hardware and methodological tools, and the resulting system quality.

The remainder of this paper is organized as follows. §2 describes our general model and two methodologies for estimating the model's parameters. §3 discusses the managerial variables selected and the empirical data collection procedures. The results obtained from estimating the model's parameters are presented and discussed in §4, the implications for managers are discussed in §5, and concluding remarks are presented in §6.

2. Conceptual Model

In developing a model of software maintenance we build upon the conceptual foundations of Kriebel and Raviv (1980) and Stabell (1982) in modeling software development as an economic production process. The input we focus on is the number of professional

[2] We will use the general term "software development" to refer to both the development of new systems and the maintenance of existing systems. "New software development" will be used to refer to situations exclusive of maintenance.

Activity	Output Measure	Input Measure
Analysis/Design	Function Points	Total Labor Hours
Coding/Testing	Source Lines of Code	

FIGURE 1. Measurement model.

labor hours, as this is the scarce constrained resource and is therefore the variable of greatest economic and managerial significance.

We model software maintenance as a complex production process involving two component activities: systems analysis/design, and coding/testing. These activities are related as illustrated in Figure 1 above. The amount of analysis/design activity required for a software maintenance project is measured by the number of Function Points, a metric developed by Albrecht at IBM in 1979 and now the standard measure of business systems functionality[3] (Albrecht and Gaffney 1983, Perry 1986). Similarly, the number of source lines of code (SLOC) is a standard measure of the size of coding/testing work (Walston and Felix 1977, Putnam 1978, Boehm 1981), an activity that gives rise to the requirement for the remainder of the professional hours on a software maintenance project.

This two component activities approach is one key distinction of our conceptual model from previous work that models software development. Previous research in this area has tended to focus on new software development, and have used either Function Points or SLOC as a surrogate for the size of the entire project, since in the case of new software development these metrics tend to be highly correlated. However, for our work in modeling software maintenance, the analysis/design and coding/testing phases need not be so tightly coupled. For example, it is easy to imagine a software maintenance project that requires a significant amount of analysis and design in order to determine the impacts of planned enhancements, but results in relatively few SLOC being added or changed. Therefore, we believe that this multidimensional approach more accurately models the special process of software maintenance.

We view software maintenance as an economic production function that relates professional labor hours incurred on a project to the size and complexity of the project measured in terms of Function Points and SLOC. Furthermore, a large stream of research has recognized that in addition to size and complexity, a number of *environmental* factors affect the labor hours required to complete a project and hence the productivity of the software maintenance effort. These factors include the ability and previous experience of the software personnel (Sackman et al., 1968, Chrysler 1978), hardware and software tools (Thadhani 1984, Lambert 1984, Boehm 1981, Jones 1986), and the attention spent on systems quality (Kriebel 1979, Mohanty 1981, Case 1985). These types of factors can also be modeled using our general approach.

Mathematically, we represent the production model as follows:

[3] Function points are a measure of the size and complexity of an *MIS*-type system. The calculation of function points involves counting the number of external input types, external output types, logical internal file types, external interface file types, and external inquiry types, and classifying them as simple, average, or complex. For example, a report that references 2 or 3 files and has between 6 and 19 data element types would count as an average external output type. The resulting weighted "function count" metric is modified by the ratings of 14 "processing complexity" factors to produce a measure of the number of Function Points. Albrecht's original exposition of the Function Points metric appears in Albrecht (1979). A revised version (and the version used in this research) appears in Albrecht and Gaffney (1983). For discussions of Function Points by other authors, see Behrens (1983), Jones (1986) and Kemerer (1987a).

$$h_j = f(y_{1j}, y_{2j}, s_{1j}, \ldots, s_{kj}) + \epsilon_j, \qquad \text{where}$$

h_j = hours for project j,

y_{1j} = Function Points for project j,

y_{2j} = SLOC for project j,

s_{ij} = environmental variable i for project j, $i = 1, \ldots, k$, and

ϵ_j = deviation from the frontier for project j.

We allow ϵ_j to take both positive and negative values. Inefficiencies in project management and implementation result in positive deviations from the production frontier, while random effects, such as errors in measuring project hours or the effects of any factors which influence productivity but are not included in the model, may cause both positive or negative deviations.[4] Given the assumption of a symmetric distribution for the deviations caused by random effects, the amount of downward deviation from the frontier will, on average, be less than or equal to the amount of upward deviations, since the upward deviations are caused by both the random effects and any production inefficiencies.

This general model can be estimated using a variety of techniques, and traditional approaches have included a number of parametric methods. In particular, we write

$$f(\mathbf{y}, \mathbf{s}) = q(y_1, y_2) + r(s_1, \ldots, s_k).$$

From an economic perspective, one question of interest is whether the function $q(y_1, y_2)$ is separable in y_1 and y_2. Our conceptual model (see Figure 1) suggests that the total labor hours spent on a project are the sum of the labor hours spent on the analysis and design activity and the labor hours incurred for coding and testing, for reasonable ranges of data as experienced in practice. Since y_1 represents analysis/design outputs (measured by Function Points) and y_2 represents coding/testing outputs (measured by SLOC), our conceptual model dictates separability of y_1 and y_2, that is

$$f(\mathbf{y}, \mathbf{s}) = q_1(y_1) + q_2(y_2) + r(s_1, \ldots, s_k).$$

For example, assuming a quadratic function, if we write

$$q(y_1, y_2) = \alpha_0 + \alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_1^2 + \alpha_4 y_2^2 + \alpha_5 y_1 y_2,$$

then the coefficient α_5 of the last term must be zero, so that $q(y_1, y_2)$ can be expressed as

$$(\alpha_0 + \alpha_1 y_1 + \alpha_3 y_1^2) + (\alpha_2 y_2 + \alpha_4 y_2^2).$$

We shall explicitly test this assumption by estimating the function and evaluating the significance of the interaction term over the range of the observed data.

In the software engineering literature (Boehm 1981, Albrecht and Gaffney 1983, Card, McGarry and Page 1987), the impact of environmental factors is assumed to be proportional to the project size. We write $s_i = z_i y_l$ where z_i is the measure of an environmental factor and $l = 1$ or 2 depending on whether the environmental factor influences the analysis/design activity or coding/testing. Therefore, we have

$$r(s_1, \ldots, s_k) = \sum_{i=1 \cdots k} \beta_i s_i = \sum_{i=1 \cdots k} \beta_i z_i y_l,$$

where β_i represents the impact of the environmental variable s_i.

[4] Aigner, Lovell and Schmidt (1977) cite a number of precedents for this interpretation, including Marschak and Andrews (1944) and Zellner, Kmenta, and Dreze (1966).

As stated above, our model of the impact of environmental variables (s_i) on labor hours reflects the fact that the effect of an environmental factor (z_i) will be proportionately greater for a larger project. For example, for an environmental factor (z_i), such as use of a manual structured analysis and design methodology, the impact on project labor hours is likely to be proportional to the level of the analysis and design activity, where the technique has its greatest impact. In such a case, our environmental variable (s_i) is therefore defined by multiplying the environmental factor (z_i) by Function Points which is the measure of the analysis and design activity. On the other hand, an environmental factor measuring machine response time will be proportional to the level of coding and testing activity, since the analysis and design activity is not dependent upon access to machine cycles.[5] The environmental variable (s_i) in this case, is defined by multiplying the factor (z_i) by SLOC, the measure of coding and testing.

More generally, we could have modeled $r(s_1, \ldots, s_k)$ to include interaction terms across various environmental factors to reflect higher order effects. We are dissuaded from doing so because of the limited number of project observations. For example, with just five environmental factors, including all the cross effects would introduce ten variables necessitating ten additional parameters to be estimated.[6]

3. Data

A. *Introduction*

We illustrate the use of our general model of software maintenance production by estimating the value of its parameters with an empirical data set collected from a site employing professional systems analysts and programmers. Since our focus is on software maintenance, we needed a site with a large inventory of existing software. In addition, this approach is believed to be more meaningful for practicing managers than the typical alternative, data from experimental results involving student programmers, who may not be representative of actual professional practice (Conte, Dunsmore and Shen 1986).

B. *Data-site and Project Selection*

The data-site for the testing of the model is a large commercial bank's information systems department (hereafter referred to as "the Bank"). 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 information systems (Freedman 1986, Withington 1987) and therefore this site is likely to be representative of much of current business information systems. The information systems 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: sufficient size and recency. Selecting larger projects allows the examination of 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 (Curtis 1981, Dickey 1981, DeMarco 1982).

[5] Of course, some of the newest approaches use automated tools to support the analysis and design phase, which are dependent upon machine cycles. However, these tools were not in use on projects at the datasite during the time period studied.

[6] However, for sensitivity analysis purposes we did rerun the model described later in the paper and found the additional higher order variables to have no significant explanatory power.

Therefore, we only considered projects at the Bank that cost a minimum of $5,000 in internal dollars.[7]

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 accurate 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 65 of 84 projects completed during that period.

C. *Variables and Data Collection*

Previous research on modeling new software development has identified over 100 possible variables to explain variations in productivity (Jones 1986). However, this work also suggests that many of these factors are site-specific, that is, factors that are critical in one environment (e.g., real-time command and control systems) may not be present in another environment (e.g., commercial information systems). Therefore, we focused our research on those variables believed to be most important by managers at the Bank. Our general model formulation supports a large number of variables. We illustrate the model here with five variables due to the limited amount of data available at this site. Given the nature of the phenomenon studied, extensive efforts were made to collect data that were both objectively measurable and under the control of managers in the information systems department. Both of these criteria were believed to be critical if the resulting model was to be perceived as both valid and useful to the managers at the Bank. In addition, special care was taken in measuring and checking these data, as described below.

Data were collected on professional (i.e., exempt) labor hours (h_j), Function Points (y_{1j}) and SLOC (y_{2j}). Only Function Points added or changed were counted, using the rules reported by Albrecht (1983). Added or changed SLOC were counted according to the rules reported by Jones (1986, p. 20). The environmental factors included the ability of the project team members, the level of previous experience with the application, the use of a structured analysis and design software development methodology, the level of hardware response time, and the operational quality of the resulting system. The key input variable was the number of work-hours charged *by person* to the project. Raw data on work hours was obtained from the Bank's project reporting system. These data are used to chargeback labor costs to requesting user departments, and are therefore believed to be of sufficient accuracy to be used in this research. Previous research has generally been satisfied with total aggregated work-hours by project only (Walston and Felix 1977, Crossman 1979, Albrecht and Gaffney 1983, and Behrens 1983). 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 modified 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. In this paper, while our unit of analysis is the project, we characterize the actual work-hours expended along a number of individual dimensions, particularly capability and experience.

[7] The mean size of these projects was 937 work-hours, with the smallest project being 130 work-hours. Therefore, very small "one-day fixes" are specifically excluded from this sample. The means for the output variables were, for Function Points, 118, and for SLOC, 5,415. The Bank data fall along a spectrum of three categories, with "type 1" being closest to new development, and "type 3" being closest to pure repairs. The distribution of projects in the sample is 20% type 1, 58% type 2, and 22% type 3 (Kemerer 1987b).

A measure of capability of individual project team members was obtained from the Bank's existing personnel system. The Bank gives annual performance appraisals, with a staff member's review being summarized into a numerical score on a scale of 1 to 5, 1 being the best. As these scores are used by the Bank for promotion and salary decisions, and are reviewed at multiple levels, they were felt to be valid indicators of ability. In addition, they had the advantages of being contemporaneously collected and being familiar to the Bank's information systems development managers. The characterization of previous experience of the team members was accomplished via a personnel data collection form that requested each project member to fill in data on his or her application experience. The forms were matched to the records in the time reporting system via an employee number.

An important issue in measuring productivity is whether the products of efficient projects are of the same quality as those of less efficient projects. The quality concept used here 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, daily abnormal end (abend) reports, weekly section status reports, and ad hoc user problem reports. Data from the two month period following implementation of each project were compared with data from the previous twelve months' trend. Projects were rated as belonging to one of three quality categories: below average, average, or above average. For example, a project is rated above average if (1) no major abends or problems specifically tied to project turnover are reported, and (2) the number of minor turnover problems is less than one standard deviation from the average number of abends or reported problems in the previous twelve months. This rating was generated by a team of three individuals within the Bank for each project. These results were then forwarded to the section heads for review and validation. The net result of this process was that, of the 65 projects, 9 were rated as below average, 43 were rated as average, and 13 were rated as above average.

Finally, data on Function Points, SLOC, use of the structured methodology, and hardware response time were captured via distribution of a data collection form to project leaders. All questions were drawn from previous research, particularly Albrecht and Gaffney (1983), Boehm (1981) and Jones (1986). In addition, 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. Further, 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 provide additional assurances of data validity. After the project leader had completed the data collection form, it was first reviewed by the section head. After review by the section head, the data collection form was forwarded to the research team for a second review for completeness, reasonableness and consistency across projects. In summary, we focused on *observable* variables deemed important by managers at the Bank, and exercised much due care to ensure that the data were accurately collected.

After collection, the environmental factors (z_i) were operationalized as follows. Since the focus of the study was on maintenance, we collected data on team member's previous experience with the application system being modified. Based on discussions with Bank information systems managers, it was felt that a possible success factor was the presence of at least one project member with significant[8] application experience. This was necessary in order that the individual's experience could be leveraged over the project team.

[8] Deemed to be two years by a consensus of managers at the Bank, a result that is also supported by the software engineering literature (e.g., see Jeffrey and Lawrence 1985). As the Bank's hiring policies strongly favor hiring graduates directly out of school, a project member's application experience tends to be highly correlated with his or her experience with the Bank's version of that system.

Therefore, z_1 is a dummy variable indicating the presence (0) or absence (1) of an experienced project team member.

Capability of the project team was viewed as a summary of the capabilities of individual team members. Using the numerical ratings from the personnel system, staff members with ratings of 1 or 2 were rated as above average capability team members, and those rated 3, 4 or 5 were rated as average capability. This apportionment resulted in two approximately equal size groups. These ratings were mapped to the hourly data and the resulting z_2 variable equaled the percentage of the hours charged to the project by the above average capability staff members.

A current topic of great interest among software development practitioners is the efficacy of new software development tools and techniques. We had the opportunity to observe the impact of one of these techniques in the form of a proprietary structured analysis and design methodology that had been purchased by the Bank in the past year, and had been used on some but not all of the projects in our sample. The variable z_3 is a dummy variable indicating the use (1) or absence (0) of this technique.

The quality of the resulting system (z_4) is a three-valued categorical variable indicating better than average, average, or less than average resulting system quality as described above (Perry 1986). Previous software engineering research has tended to focus on either productivity or quality to the exclusion of the other. We simultaneously investigate both of these phenomena in our analysis.

As noted earlier, numerous previous researchers have demonstrated the importance of hardware response time in system developer productivity. We used the measure developed by Boehm (1981) to characterize online response time and batch turnaround time of less than four hours as good response. The variable z_5 is a dummy variable indicating the presence (1) or absence (0) of good response time.

While we have chosen environmental factors that are globally supported in the literature and recognized locally by the managers at the Bank, our research is novel in going beyond the "black-box" approach to software development and decomposing the maintenance activity into its two component parts. Therefore, this requires a model formulation that specifies which of the compound activities is believed to be more greatly impacted by each of the environmental factors.

The first three z_i environmental factors were multiplied by y_1, Function Points. The factors z_1 and z_2 represent the impact of team member capability and previous experience with the application, and it was believed that their impact would be most strongly felt during analysis/design, where the greatest amount of leverage from their capability and experience would be obtained. The third factor, z_3, representing the use of a structured analysis and design methodology, clearly is associated with the analysis and design output, y_1. The last two z_i factors operational quality (z_4) and good response time (z_5), were believed to most strongly impact the coding/testing phase. The errors represented by operational quality, are of the type reflecting poor coding technique and/or insufficient testing. Response time clearly is more directly related to coding/testing activities than analysis/design work that is typically not dependent upon access to machine cycles.

4. Model Estimation

In this section we estimate a stochastic production frontier based upon the model developed in §2. In the first part of this section, we use a parametric formulation. In the second part we show that similar results are obtained when a nonparametric approach is used. We discuss the managerial implications of the results in §5.

A. *Parametric Formulation*

The estimation of a stochastic frontier requires the specification of a distribution for the error term. In particular, the econometric modeling of production frontier considers

the error term ϵ to be composed of two terms, η and ν, where η represents the impact of random effects and ν represents the impact of production inefficiency.

As in Aigner, Lovell & Schmidt (1977), hereafter ALS, we assume the following error structure: $\epsilon = \eta + \nu$, where η is distributed as $N(0, \sigma_\eta^2)$ and ν is distributed as a half-normal $|N(0, \sigma_\nu^2)|$.

Following Weinstein (1964), the density function of ϵ, the composed error, is given by

$$f(\epsilon) = (2/\sigma)f^*(\epsilon/\sigma)[F^*(\epsilon\lambda/\sigma)], \qquad -\infty \le \epsilon \le +\infty,$$

where $\sigma^2 = \sigma_\eta^2 + \sigma_\nu^2$ and $\lambda = \sigma_\nu/\sigma_\eta$ and $f^*(\cdot)$ and $F^*(\cdot)$ are the standard normal density and distribution functions, respectively. ALS estimate the parameters using the maximum likelihood estimation (MLE) method.

The specific parametric form adopted for the nonlinear function $q(y_1, y_2)$ is

$$q(y_1, y_2) = \alpha_0 + \alpha_1 y_1 + \alpha_2 y_2 + \alpha_3 y_1^2 + \alpha_4 y_2^2 + \alpha_5 y_1 y_2.$$

To test whether $q(y_1, y_2)$ is a separable production function of the form $q_1(y_1) + q_2(y_2)$, as may be the case in software maintenance, we need to test the hypothesis that $\alpha_5 = 0$.

Combining this form of the function $q(y_1, y_2)$ with the form of the function $r(s_1, \ldots, s_k)$ developed in §3 and the composed error formulation described above, the estimable form of the model is

$$f(\mathbf{y}_j, \mathbf{s}_j) = f'(\mathbf{y}_j, \mathbf{z}_j) = \alpha_0 + \alpha_1 y_{1j} + \alpha_2 y_{2j} + \alpha_3 y_{1j}^2 + \alpha_4 y_{2j}^2 + \alpha_5 y_{1j} y_{2j} + \beta_1 z_{1j} y_{1j} + \beta_2 z_{2j} y_{1j} + \beta_3 z_{3j} y_{1j} + \beta_4 z_{4j} y_{2j} + \beta_5 z_{5j} y_{2j} + \eta_j + \nu_j$$

for all $j = 1, \ldots, 65$.

The maximum likelihood estimates of the parameters of this model are shown in Table 1. The maximum likelihood estimate of σ_η is 351.4, while that of σ_ν is 0, which means that all of the deviations from the estimated production frontier are represented by random effects, and none due to inefficiency. This is, of course, predicated upon the given

TABLE 1

MLE *Results, Quadratic Formulation* $R^2 = .80$, *adj.* $R^2 = .76$, *F-stat.* = 21.303 (.000)

Parameter	Coefficient	Standard Error	*t*-Ratio	(Significance)
α_0	444.151	83.610	5.312	(.000)
α_1	.169	1.957	.086	(.931)
α_2	.145	.033	4.438	(.000)
α_3	.009	.003	2.643	(.008)
α_4	−.000	.000	−1.733	(.083)
α_5	−.000	.000	−.634	(.526)
β_1	.527	.639	.825	(.409)
β_2	−.027	.017	−1.583	(.113)
β_3	2.173	.681	3.190	(.001)
β_4	−.015	.009	−1.648	(.099)
β_5	−.031	.020	−1.569	(.117)

specification of the model, with the inclusion of the environmental variables, s_i. Since $\sigma_\nu = 0$, the estimation results correspond to ordinary least squares estimates in this case.[9]

An F-test of α_3, α_4 and α_5, the coefficients of the quadratic terms in the $q(y_1, y_2)$ function, being all equal to zero was rejected at the 5 percent level, indicating that there exist possible economies and diseconomies of scale in software maintenance. This is consistent with the previous results of Banker and Kemerer (1989) with respect to new software development, and suggests that linear models are likely to be inadequate representations of the production process. The estimate of α_5 is not significantly different from zero at conventional levels of significance, supporting the notion that, for the relevant range exhibited by these data, the function $q(y_1, y_2)$ is separable in y_1 and y_2. Given that the modeling of software development as an economic production function with multiple outputs has not been widely developed in the literature, this appears to be the first empirical test of the separability hypothesis. One possible interpretation of these results is that the separability of the analysis/design and coding/testing activities may hold for a larger range for maintenance than for new development. This is because activities of the project team in both phases are likely to be far more constrained in a maintenance context by the presence of the existing system. Further research seems indicated to explore whether this result would also hold for new software development, and of course to validate this result for maintenance at other datasites.

B. *SDEA Estimation*

Although the parametric stochastic estimation of production frontiers described above models the impact of both inefficiency and random effects, it does assume a particular functional form for the production correspondence and the error structure. The statistical distributions of the estimated parameters are based on these assumed functional forms. Inferences from the statistical tests are consequently conditional on the parametric specification of the model correctly reflecting the underlying production relation (see Hildenbrand 1981, Varian 1984, and Banker and Maindiratta 1988). But the choice of a particular functional form can be difficult to justify on a priori grounds. This problem can be partially mitigated by using flexible functional forms such as the translog that can be used to approximate the unknown true production function. Unfortunately, these forms require the estimation of a large number of parameters relative to the available observations. Furthermore, the underlying regularity conditions of monotonicity and strict quasi-concavity are often violated at many points of most data sets, thus biasing inference; for a theoretical analysis of regularity conditions, see Caves and Christensen (1980).[10]

The problems inherent in parametric estimation can be overcome by estimating a nonparametric stochastic frontier using the approach of Stochastic Data Envelopment Analysis (SDEA), a nonparametric method for evaluating productivity which (1) assumes only the regularity condition of monotonicity of the production function and convexity of the production possibility set, and (2) allows for the possibility of two-sided random errors in model specification or measurement in addition to the one-sided deviations attributable to inefficiency in the use of input resources (Banker, Charnes, and Cooper 1984 and Banker 1990).

[9] Goldfeld-Quandt tests did not indicate even mild heteroskedasticity, and Belsey-Kuh-Welsch collinearity diagnostics indicated an absence of a collinearity problem except for the terms y_1 and y_1^2.

[10] As a test of the sensitivity of these results, we also estimated a translog form of the model, as follows:

$$\ln(hJ) = \alpha_0 + \alpha_1(\ln y_{1j}) + \alpha_2(\ln y_{2j}) + \alpha_3(\ln y_{1j})^2 + \alpha_4(\ln y_{2j})^2 + \alpha_5(\ln y_{1j})(\ln y_{2j})$$
$$+ \beta_1(z_{1j}) + \beta_2(\ln z_{2j}) + \beta_3(z_{3j}) + \beta_4(\ln z_{4j}) + \beta_5(z_{5j}) + \epsilon_j,$$

where ϵ_j is specified as a composed error term as in the quadratic model. The signs of the estimated coefficients of the environmental factors are identical to those presented in Table 1. See also Banker, Conrad, and Strauss (1986) for alternative translog cost function formulations and comparisons with DEA estimates.

The composed error econometric models discussed above require an a priori specification of the parametric distributions of the inefficiency and the random effect components of the error term. On the other hand, the linear programming-based formulation of SDEA is specified in terms of deviations above and below the frontier, and it requires the relative weight for these positive and negative deviations to be specified in the objective function. By varying the relative weights, we examine the sensitivity of the estimation results to the postulated importance of deviations due to inefficiency or random factors.

Mathematically, the SDEA formulation of the software maintenance productivity model, assuming separability, is as follows:[11]

$$\min \sum_{j=1}^{65} cv_j + (1 - c)u_j$$

s.t.

$$\text{(i)} \quad h_j = \sum_{l=1}^{2} t_{lj} + \beta_1 z_{1j} y_{1j} + \beta_2 z_{2j} y_{1j} + \beta_3 z_{3j} y_{1j} + \beta_4 z_{4j} y_{2j} + \beta_5 z_{5j} y_{2j} + v_j - u_j \qquad \forall j,$$

$$\text{(ii)} \quad 0 \le w_{lj}(y_{lj} - y_{lk}) - (t_{lj} - t_{lk}) \qquad \forall l, j, k,$$

$$\text{(iii)} \quad t_{lj}, w_{lj}, u_j, v_j \ge 0, \beta_l \qquad \text{unconstrained.}$$

The explanation of this formulation is as follows. The objective function is a minimization of the weighted sum of the deviations from the production frontier. The u_j's represent downward deviations (due only to random effects). The v_j's are upward deviations (due to either random effects or inefficiencies, or both). Here we assume that $q(y_1, y_2) = q_1(y_1) + q_2(y_2)$ where each q_l, $l = 1, 2$, is monotonically increasing and convex. We write $t_{lj} = q_l(y_{lj})$ and let $t_j = t_{1j} + t_{2j}$. Constraint (i) is the functional relationship between the input variable labor hours on the left-hand side and the output and environmental variables on the right-hand side. Constraint (ii) ensures that the functions $t_l = q_l(y_l)$ are monotonically increasing and convex, standard economic production function assumptions. Standard nonnegativity constraints are in (iii), except that the signs of the β_l's are unconstrained.

In this extension of the standard SDEA model, we have represented the function $r(s_1, \ldots, s_k)$ of the environmental variables parametrically as in the earlier sections; and therefore we obtain summary estimates of the impact of the environmental variables. Thus the results of the SDEA formulation can be compared directly to those obtained from the previous parametric formulations.

We allowed c to vary between zero and 0.5. Values of c close to zero cause all observed deviations to be attributed to inefficiency, i.e., the usual DEA model. A value of $c = 0.5$ causes all observed deviations to be attributed to random effects. Intermediate values of c (between 0 and 0.5) cause the deviations to be attributed partly to inefficiencies and partly to random effects.

Table 2 shows the estimates of the coefficients of the environmental variables obtained using the separable SDEA formulation. From Table 2 we can observe a number of characteristics of the estimated coefficients of the environmental variables. First we see that the signs of the β_l are identical and magnitudes very similar to those obtained from the parametric formulations presented in Table 1. Our conclusions from the stochastic

[11] The formulation for the nonseparable case is given in Appendix A. An examination of the results in Appendix A shows the marked similarity between the results of both this separable model and the nonseparable model presented in the appendix. This indicates that our insights about factors affecting productivity are consistent across models that do and do not assume separability of Function Points and SLOC in estimating labor hours.

TABLE 2
Separable SDEA: Environmental Variables Coefficient Estimates

c	0.01	0.1	0.3	0.5
β_1	+0.676	+0.691	+1.114	+0.463
β_2	−0.017	−0.016	−0.025	−0.008
β_3	+2.189	+2.127	+2.043	+1.932
β_4	−0.002	−0.003	−0.001	−0.014
β_5	−0.030	−0.031	−0.065	−0.071

nonparametric analysis are thus consistent with the results of the parametric formulation. Discussion of the managerial interpretations of these results will be presented in §5.

The signs of the β_i's in Table 2 are invariant to different values of c, and their magnitudes are roughly similar. The insights that we thus obtain about the direction and magnitude of the factors that affect productivity are consistent for the different versions of the SDEA models estimated. In particular, the insights do not change with the weights on the inefficiency and random effects parameters. Appendix A further shows that these insights are unaltered if we assume nonseparability of Function Points and SLOC.

Our extension of the SDEA methodology enables us to determine the relative explanatory power of each of the environmental variables. This is accomplished by running the model $(n + 1)$ times, where n (=5 here) is the number of environmental variables. The first run includes all of the environmental variables. Then the model is rerun, omitting a single (different) variable each time. The resulting objective function values are compared by examining the ratio of the objective function value when all environmental variables are included to the objective function value when one variable is omitted in turn. This ratio shows the relative loss in explanatory power through the omission of a variable. This is analogous to the F-Test or the likelihood ratio test in econometric theory, although statistical properties are not obtained as no distributional assumptions are made about the deviations. The objective function values were computed and the ratios of the objective functions with and without suppressing each of the z variables were computed and are shown in Table 3.

The impacts of individual factors are discussed in §5. We simply note here that Table 3 reveals that the factors with the greatest impact are z_3 (use of structured methodology) and z_5 (good response time). Omitting these two factors most affects the weighted sum

TABLE 3
Separable SDEA: Percentage Increase in Objective Function Values on Suppressing Environmental Variables

c	$c = 0.01$	$c = 0.10$	$c = 0.30$	$c = 0.50$
z_1 suppressed	2.85%	3.13%	3.02%	0.37%
z_2 suppressed	2.00%	0.87%	1.95%	0.76%
z_3 suppressed	23.03%	21.00%	19.75%	10.44%
z_4 suppressed	4.08%	4.09%	2.58%	3.12%
z_5 suppressed	8.82%	8.31%	8.77%	10.92%

TABLE 4
Percentage Increase in Objective Function Values on Imposing Separability

c	$c = 0.01$	$c = 0.10$	$c = 0.30$	$c = 0.50$
No z suppressed	5.57%	4.75%	3.69%	4.62%

of absolute deviations. The positive sign on the coefficient of the structured methodology variable (see value of β_3 in Table 2) indicates that the use of the structured methodology for software maintenance increases labor hours incurred on analysis and design activities, a result that is discussed in some detail in §5. The negative sign on the good response time coefficient (see value of β_5 in Tables 1 and 2) suggests that good response time has the effect of reducing labor hours spent on coding and testing.

Table 4 indicates the extent to which the objective functions values are higher when the separability assumption is imposed in the SDEA model. The percentage increase in the objective function value of the order of 4–5% for various values of c supports the notion that the impacts of Function Points and SLOC on labor hours are largely separable in the context of software maintenance for the range observed at this datasite. This is not a critical issue here, since the estimated sign and magnitudes of the coefficients of the environmental variables, the principal focus of this paper, are similar in the separable and nonseparable cases.

5. Implications for Software Maintenance Management

Our model provides a number of insights for the management of software maintenance. An important benefit from using such a model is that it allows a quantifiable estimate of the impact on productivity of a number of environmental variables under managerial control. For example, Table 1 indicates that the most significant factor was the negative impact upon short-term productivity of the use of the structured analysis and design methodology (β_3). A managerial interpretation of the data in Table 1 is that for each Function Point, the use of the methodology adds 2.17 labor hours to the project. As the mean number of Function Points in this dataset is 118, this suggests that the average impact of the methodology is to increase this number of labor hours by 256. (The β_3 coefficient estimation using the nonparametric SDEA model also yields very similar results. See Tables 2 and A.1.) This result makes intuitive sense in that many of the presumed benefits of using a detailed methodology that requires a lot of documentation are not observed until the follow-on projects, when enhancements or repairs need to be made to the system. In the short term, as measured in this research, 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 noted that use of this methodology was new at the Bank. Therefore, project team members were typically using this methodology for the first time, and thus these results may be a manifestation of a learning curve phenomenon at the Bank.

Previous research investigating the productivity effects of various software engineering management variables involving homogeneous datasets (i.e., data from similar project types at the same firm) have typically reported results with low statistical significance. For example, Kemerer (1987a) reported that the addition of the large numbers of 'productivity factors,' common in widely-used software cost estimation models, did little to improve the relationship between size and effort on a set of 15 homogeneous MIS-type projects. Card, McGarry and Page (1987), in a study 22 spacecraft flight dynamics programs at the Software Engineering Laboratory, found that *none* of the eight technologies investigated had a statistically significant effect on productivity at even a 15% level. One reason for these findings is perhaps the lack of inherent variability in the data. Despite

this inherent difficulty, in the parametric analysis of software maintenance productivity developed in §2 and estimated in §4, the managerial variables had signs in the expected direction, and were significant at the following levels: structured methodology (β_3), 0.1%, higher quality (β_4), 9.9%, higher ability staff (β_2), 11.3%, good response time (β_5), 11.7%, and absence of application experience (β_1), 40.9%, although only the null hypothesis that the coefficient of the structured methodology variable being equal to zero was rejected at the conventional 5% level for a two-tailed test.

Our results also suggest that a relatively small number of critical factors may explain a large amount of the variation in productivity at a specific site. Although we believe that the basic insights from our analysis will carryover to other similar Cobol-oriented environments, an implication for practicing software maintenance managers is that they should attempt to capture data on projects at their own site, as the relatively small number of critical factors may vary across sites. We would suggest modeling software maintenance as a multi-output production function, as was done in the current research. In order to assist managers in performing such analyses, the following guide to interpreting some of the less obvious results is presented, using the "higher ability staff" (β_2) variable for our example. The coefficient for β_2 is estimated by the SDEA model to be approximately $-.02$ (see Table 2). An interpretation of this is that the impact of substituting higher ability for lower ability staff members would be to reduce the number of hours by .02 per Function Point per percentage of the hours substituted. To illustrate this with two actual values from the observed range in the dataset, take the case of an average project (118 Function Points) staffed with only ten percent above average ability team members. If we were to substitute a full (100%) above average staff, then we would estimate the difference to be 212 ($=90*.02*118$) fewer labor hours charged on an average project.

It is also interesting to compare these results with the maintenance version of Boehm's well known COCOMO model (Boehm 1981). The maintenance COCOMO model is essentially equivalent to the standard COCOMO model with its 15 cost drivers, except that it: (1) allows for the notion of changed code rather than all code as in new development (Boehm 1981, p. 71), (2) eliminates the SCED cost driver, and (3) makes small changes to the weights assigned to the RELY and MODP cost drivers (Boehm 1981, pp. 129–130).

In our model we also adopt the notion of changed functionality and include five "cost driver" type variables, which were chosen based on our software development experience and their relevance to the data-site. In comparing our variables to the variables in maintenance COCOMO an approximate correspondence would be as follows: The COCOMO variable AEXP is represented by our z_1 application experience variable, the COCOMO variables ACAP and PCAP are represented in the Bank environment by our z_2 capability variable, the COCOMO variables TOOL and MODP are approximately correspondent with our z_3 structured analysis and design variable, the COCOMO variable TURN corresponds to our z_5 response time variable. Our z_4 quality variable is not represented in COCOMO. Other COCOMO variables were excluded because they were either (a) specific to the primarily real-time/embedded systems studied by Boehm in his TRW database (e.g., TIME, STOR, RELY), (b) relatively homogeneous across Bank projects (e.g., DATA, VIRT), or (c) captured in the Bank environment by our other variables (e.g., CPLX by y_1, VEXP and LEXP by z_1).

Our results for greater capability (positive influence on productivity), and better response time (positive influence on productivity) are consistent with those of Boehm (see Boehm 1981, p. 118). However, even though our structured analysis and design variable does not exactly correspond with COCOMO's TOOL and MODP variables, the fact that it exerts a negative influence, while COCOMO might posit a positive influence, strongly suggests the need for more research in this area, perhaps in the form of laboratory studies, to determine under what conditions each of the results might be expected.

Due to the novelty of the structured analysis and design result, we conducted further

sensitivity analysis on it. Of course, the negative result of the Belsley-Kuh-Welch test of collinearity mentioned earlier indicates that the results are not confounded by collinearity problems. More specifically, we tested for possible correlations between the use of the structured analysis and design methodology and other factors, such as project size and complexity or quality. The Pearson correlation of the structured methodology factor with Function Points was .11, with SLOC, .05, and with the quality factor was −.13. None of these correlations are significantly different than zero at even the 30% level.[12] These results support the contention that the negative results for the structured methodology are not confounded by their use on either large (small) projects or projects resulting in higher (lower) quality.

In general, little research is available on the productivity impacts of structured analysis and design methodologies. Their assumed net positive impact may not be obtained immediately, just as Vessey and Weber (1984) found only weak support for the positive impacts of structured programming. Further research needs to be done to validate these results, and longitudinal studies should be performed to determine any possible differential long run effects.

6. Concluding Remarks

In this paper we have provided a new model of software maintenance productivity. This model is a significant advance over previous work in software productivity in that it (a) provides for the multi-dimensionality of software lifecycle products, (b) allows and tests for possible nonlinearities and separability of the software production function, (c) explicitly models individual project member differences, and (d) directly addresses software maintenance (rather than new development), an activity of increasing economic importance and managerial concern.

In developing methods to estimate the model we have extended the Stochastic Data Envelopment Analysis approach to allow the simultaneous consideration of inputs, outputs, and other factors. This allows the evaluation of the marginal impact of each of the factors influencing productivity. This general approach can be used to evaluate productivity in a wide array of managerial contexts.

The use of this model is illustrated using actual data from a commercial bank. While the external validity of the results on the impact of specific factors will need to be supported with results from other sites, our choice of COBOL transaction processing systems is one that is representative of the majority of business information systems being maintained today, and therefore is of immediate practical relevance. The availability of larger datasets could allow for the inclusion of a greater number of productivity influencing factors, as our general model in its current form is not limited to the five shown in the illustrative example. We have also done extensive sensitivity analysis of our results relative to alternative forms of the production function and its estimation, and find them to be very consistent. This is a useful first step in examining an area that is of considerable importance in the management of information systems.[13]

[12] We also computed the *contingency coefficient* version of these correlations (Siegel 1956, pp. 196–202). None of these correlations were significantly different than zero (based on the chi-square test) at even the 20% level.

[13] We gratefully acknowledge research support from the National Science Foundation Grant No. SES-8709044, the Center for the Management of Technology and Information in Organizations (Carnegie Mellon University), the International Financial Services Research Center (MIT), and the Center for Information Systems Research (MIT). The cooperation of managers at the data-site was invaluable. Helpful comments were provided by participants in research seminars at Carnegie Mellon University, Georgia State University, University of Michigan, University of Minnesota, MIT, Rensselaer Polytechnic Institute, University of Rochester, UCLA and the Wharton School at the University of Pennsylvania. Presentations at the EURO-TIMS Joint International Conference and the International Conference on Information Systems generated useful feedback. Helpful comments from four anonymous referees and the associate editor are also gratefully acknowledged.

Appendix A. Sensitivity Analysis of the SDEA Model to the Separability Assumption

In this appendix, we present an alternative SDEA model that does not assume separability of the production function. Mathematically, this model is as follows:

$$\min \sum_{j=1}^{65} cv_j + (1-c)u_j$$

s.t.

(i) $h_j = t_j + \beta_1 z_{1j} y_{1j} + \beta_2 z_{2j} y_{1j} + \beta_3 z_{3j} y_{1j} + \beta_4 z_{4j} y_{2j} + \beta_5 z_{5j} y_{2j} + v_j - u_j \qquad \forall j,$

(ii) $0 \le \sum_{l=1}^{2} w_{lj}(y_{lj} - y_{lk}) - (t_j - t_k) \qquad \forall j, k,$

(iii) $t_j, w_{lj}, u_j, v_j \ge 0; \quad \beta_i$ unconstrained, where $t_j = q(y_{1j}, y_{2j})$.

This model has the same objective function as the separable model presented in §4B, and only slightly modified constraints. The results from this model are presented in Table A.1 below. Note that the values are essentially equivalent to those obtained under the assumption of separability. Carrying the parallel analysis forward, Table A.2 presents the equivalent data to the separable model's Table 3.

TABLE A.1

Nonseparable SDEA: Environmental Variables Coefficient Estimates

c	0.01	0.1	0.3	0.5
β_1	+0.703	+0.670	+0.443	+0.215
β_2	−0.011	−0.007	−0.014	−0.010
β_3	+1.891	+1.971	+2.304	+1.914
β_4	−0.007	−0.006	−0.006	−0.006
β_5	−0.038	−0.041	−0.052	−0.069

TABLE A.2

Nonseparable SDEA: Percentage Increase in Objective Function Values on Suppressing Environmental Variables

c	$c = 0.01$	$c = 0.10$	$c = 0.30$	$c = 0.50$
z_1 suppressed	1.99%	1.78%	1.14%	0.13%
z_2 suppressed	1.13%	0.60%	0.91%	0.72%
z_3 suppressed	12.53%	11.41%	12.05%	9.51%
z_4 suppressed	6.34%	5.96%	4.04%	6.40%
z_5 suppressed	9.83%	9.27%	8.74%	9.45%

As can be seen by comparing Tables 3 and A.2, the managerial insights regarding the impact of each of the five managerial variables is relatively insensitive to the separability assumption. Thus, we can conclude that the effects of these variables are not materially changed no matter which assumption is made about the nature of the production function.

References

ADAMS, E. N., "Optimizing Preventive Service of Software Products," *IBM Res. J.*, 28, 1 (1984).

AIGNER, D., C. LOVELL AND P. SCHMIDT, "Formulation and Estimation of Stochastic Frontier Production Function Methods," *J. Econometrics*, 6 (1977), 21–37.

ALBRECHT, A., "Measuring Application Development Productivity," *Proc. IBM Appl. Development Sympos., GUIDE/SHARE*, (October 1979), 83–92.

——— AND J. GAFFNEY, JR., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Trans. on Software Engineering*, SE-9, 6 (November 1983), 639–648.

BANKER, R., "Stochastic Data Envelopment Analysis," *Management Sci.*, (1990), to appear.

———, A. CHARNES AND W. COOPER, "Some Models for Estimating Technical and Scale Inefficiencies in DEA," *Management Sci.*, 30, 9 (September 1984), 1078–1092.

———, R. CONRAD AND R. STRAUSS, "Comparative Application of Data Envelopment Analysis and Translog Methods," *Management Sci.*, 32, 1 (January 1986), 30–43.

———, S. DATAR AND C. KEMERER, "Factors Affecting Software Maintenance Productivity: An Exploratory Study," *Proc. 8th International Conf. on Information Systems*, Pittsburgh, PA, (December 1987), 160–175.

——— AND C. KEMERER, "Economies of Scale in New Software Development," *IEEE Trans. on Software Engineering*, 15, 10 (October 1989).

——— AND A. MAINDIRATTA, "Nonparametric Analysis of Technical and Allocative Efficiencies in Production," *Econometrica*, (November 1988), 1315–1332.

BEHRENS, C., "Measuring the Productivity of Computer Systems Development Activities with Function Points," *IEEE Trans. on Software Engineering*, SE-9, 6 (November 1983), 648–652.

BELADY, L. A. AND M. M. LEHMAN, "Evolution Dynamics of Large Programs," *IBM Systems J.*, 15, 1 (1976).

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

CARD, D., F. MCGARRY AND G. PAGE, "Evaluating Software Engineering Technologies," *IEEE Trans. on Software Engineering*, SE-13, 7 (July 1987), 845–851.

CASE, A., "Computer-Aided Software Engineering," *Database*, 17, 1 (Fall 1985), 35–43.

CAVES, D. AND L. CHRISTENSEN, "Global Properties of Flexible Functional Forms, " *American Econ. Rev.*, (1980), 422–432.

CHARNES, A., W. COOPER AND E. RHODES, "Measuring the Efficiency of Decision Making Units," *European J. Oper. Res.*, 2, 6 (1978), 429–444.

———, ——— AND ———, "Evaluating Program and Managerial Efficiency: An Application of Data Envelopment Analysis to Program Follow Through," *Management Sci.*, 27, 6 (June 1981), 668–697.

CHRYSLER, E., "Some Basic Determinants of Computer Programming Productivity," *Comm. ACM*, 21, 6 (June 1978), 472–483.

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

CROSSMAN, T., "Taking the Measure of Programmer Productivity," *Datamation*, 25, 5 (May 1979), 144–147.

CURTIS, B., "Substantiating Programmer Variability," *Proc. IEEE*, 69, 7 (July 1981), 846.

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

DICKEY, T., "Programmer Variability," *Proc. IEEE*, 69, 7 (July 1981), 844–845.

DICKSON, G., R. LEITHEISER AND J. C. WETHERBE, "Key Information Systems Issues for the 1980's" *MIS Quart.*, 8, 3 (September 1984), 135–159.

ELSHOFF, J., "An Analysis of Some Commercial PL/1 Programs," *IEEE Trans. on Software Engineering*, SE2, 2 (1976), 113–120.

FREEDMAN, D., "Programming without Tears," *High Technology*, 6, 4 (April 1986), 38–45.

GALLANT, J., "Survey Finds Maintenance Problem Still Escalating," *Computerworld*, 20, 4 (January 1986), 31.

HILDENBRAND, W., "Short-Run Production Functions Based on Microdata," *Econometrica*, 49, 5 (September 1981), 1095–1125.

JEFFERY, D. AND M. LAWRENCE, "Managing Programming Productivity," *Systems and Software*, 5 (1985), 49–58.

JONES, C., *Programming Productivity*, McGraw-Hill, New York, 1986.

KEMERER, C., "An Empirical Validation of Software Cost Estimation Models," *Comm. ACM*, 30, 5 (May 1987a), 416–429.

———, "Measurement of Software Development Productivity," unpublished Ph.D. dissertation, UMI #8905252, Carnegie Mellon University, Pittsburgh, PA, 1987b.

KOLODZIEJ, S., "Gaining Control of Maintenance," *Computerworld Focus*, 20, 7A (February 19, 1986), 31–36.

KRIEBEL, C., "Evaluating the Quality of Information Systems, " in *Design and Implementation of Computer Based Information Systems*, Chapter 2, Sitjhoff & Noordhoff, The Netherlands, 1979, 29–43.

——— AND A. RAVIV, "An Economics Approach to Modeling the Productivity of Computer Systems," *Management Sci.*, 26, 3 (March 1980), 297–311.

LAMBERT, G. N., "A Comparative Study of System Response Time on Program Developer Productivity," *IBM Systems J* , 23, 1 (1984), 36–43.

LIENTZ, B. AND E. SWANSON, *Software Maintenance Management*, Addison-Wesley, Reading, MA, 1980.

MARSCHAK, J. AND W. ANDREWS, "Random Simultaneous Equations and the Theory of Production," *Econometrica*, 12 (July–October 1944), 143–205.

MEEUSEN, W. AND J. VAN DER BROECK, "Efficiency Estimation from Cobb-Douglas Production Functions with composed Error," *International Economic Rev.*, (1977), 435–444.

MELLOR, P., "Modelling Software Support," *ICL Tech. J.*, (November 1983), 407–431.

MOHANTY, S., "Software Cost Estimation: Present and Future," *Software-Practice and Experience*, 11 (1981), 103–121.

PERRY, W., "The Best Measures for Measuring Data Processing Quality and Productivity," Technical Report, Quality Assurance Institute, 1986.

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

SACKMAN, H., W. ERIKSON AND E. GRANT, "Exploratory Experimental Studies Comparing Online and Offline Programming Performance," *Comm. ACM*, 11, 1 (January 1968), 3–11.

SIEGEL, S., *Nonparametric Statistics for the Behavioral Sciences*, McGraw-Hill, NY, 1956.

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

THADHANI, A., "Factors Affecting Productivity During Application Developement," *IBM Systems J.*, 23 (1984), 19–35.

VARIAN, H., "The Nonparametric Approach to Production Analysis," *Econometrica*, 52, 3 (1984), 579–597.

VESSEY, I. AND R. WEBER, "Some Factors Affecting Program Repair Maintenance: An Empirical Study," *Comm. ACM*, 26, 2 (February 1983), 128–134.

——— AND ———, "Research on Structured Programming: An Empiricist's Evaluation," *IEEE Trans. on Software Engineering*, SE-10, 4 (July 1984), 397–407.

WALSTON, C. AND C. FELIX, "A Method of Programming Measurement and Estimation," *IBM Systems J.*, 16, 1 (1977), 54–73.

WEINSTEIN, M., "The Sum of Values from a Normal and a Truncated Normal Distribution," *Technometrics*, 6, 1 (February 1964), 104–105.

WITHINGTON, F., "The Pressure's On," *Datamation*, 33, 1 (January 1, 1987), 53–55.

ZELLNER, A., J. KMENTA AND J. DREZE, "Specification and Estimation of Cobb-Douglas Production Function Models," *Econometrica*, 34, 4 (October 1966), 784–795.