An Empirical Analysis of Productivity and Quality in Software Products

M. S. Krishnan
University of Michigan Business School
Ann Arbor MI 48109

C. H. Kriebel
Sunder Kekre
Tridas Mukhopadhyay
Graduate School of Industrial Administration
Carnegie Mellon University, Pittsburgh PA 15213

November 1996
Revised May 1998
Revised April 1999

We wish to thank Prof. Chris Kemerer; Prof. Barry Boehm; and seminar participants at Carnegie Mellon University, Software Engineering Institute, Workshop in Information Systems Economics-1995; University of Maryland; Stanford University; Georgia Institute of Technology; University of Texas at Austin; Case Western Reserve University; Pennsylvania State University; University of Michigan; and the 1996 Software Engineering Process Group conference, for their useful suggestions and comments.

We are grateful to John Botsford and Karen Bennet for their support and valuable suggestions. Financial support for this study was provided through a fellowship from the Software Engineering Institute, Carnegie Mellon University.

An Empirical Analysis of Productivity and Quality in Software Products

Abstract

We examine the relationship between life-cycle productivity and conformance quality in software products. The effects of product size, personnel capability, software process, usage of tools and higher front-end investments on productivity and conformance quality were analyzed to derive managerial implications based on primary data collected on commercial software projects from a leading vendor. Our key findings are as follows. First, our results provide evidence for significant increases in life-cycle productivity from improved conformance quality in software products shipped to the customers. Given that the expenditure on computer software has been growing over the last few decades, empirical evidence for cost savings through quality improvement is a significant contribution to the literature. Second, our study identifies several quality drivers in software products. Our findings indicate that higher personnel capability, deployment of resources in initial stages of product development (especially design) and improvements in software development process factors are associated with higher quality products.

1. Introduction

Computer software has emerged as a major worldwide industry, with an estimated annual budget exceeding $370 billion [Keil 1995]. It is now widely recognized that computer software accounts for a significant share of corporate information systems budgets [Humphrey, 1992]. Most corporate information systems depend on computer software for accurate and timely information. Software is also viewed as an important corporate asset in the 1990s. Yet, the software community has long been faced with severe difficulties in delivering and supporting quality software products, on time [Blackburn, Scudder and van Wassenhove, 1996a]. Additionally, life-cycle costs of software projects often turn out to be enormous and significantly over budget.

In order to overcome these difficulties, quality management and related principles (e.g., continuous process improvements and process management) have been applied to the development

and maintenance of software [Curtis, et. al, 1992]. Despite such efforts, the software industry continues to be plagued by an inability to develop quality software products [DeMarco, 1995]. In many organizations, productivity and schedules for software projects are largely unpredictable, and product quality is often poor. Software vendors are therefore attempting to build quality into the product by avoiding defects in the first place instead of removing the defects in the product through rigorous testing. International standards for quality such as ISO-9000-3 and the Capability Maturity Model (CMM) of the Software Engineering Institute have been adopted by many software organizations over the past few years [Paulk, et al., 1993a; Humphrey, 1992]. However, the effect of these new software development processes on productivity or quality has not been empirically tested. Thus, from a research perspective, it is important to identify the drivers of productivity and quality, and establish the relationship between productivity and quality.

Our field study is based on primary data collected on commercial software projects of a leading vendor. We analyze the drivers of quality and productivity such as personnel capability, product size, software process factors and usage of tools. We test the efficacy of improved processes and up-front investment in quality on life-cycle productivity of the projects. Our findings indicate significant increases in life-cycle productivity with improvements in quality. Managerial implications of our results are twofold. First, our results enable software product managers to assess cost savings from reducing the defects in their products shipped. For example, we find improving quality by 1% leads to a 0.56% gain in life-cycle productivity. Second, our results provide managers with guidelines for resource deployment during product development. For instance, higher investments in the front-end of the product development cycle leads to higher quality.

The rest of the paper is organized as follows. In the next section, we briefly review the literature and highlight the contributions of our field study. We address the research issues in Section 3 and provide a theoretical basis for our empirical models in Section 4. We describe the data

collection methodology and variable measures in Section 5. Our empirical analysis is described in Section 6, the results in Section 7. Finally, we present conclusions and limitations of our study in Section 8.

2. Literature Review and Contributions

Identifying software productivity factors and estimating software costs continue to be important research topics [Kemerer, 1987; Mukhopadhyay and Kekre, 1992; Banker et al., 1993]. Recently, Maxwell, et al., [1999] have reported generic and company specific models to identify software development cost drivers. Most of the empirical research on software maintenance has analyzed tradeoffs between software quality and maintenance effort, and identified drivers of software maintenance costs. Theoretical models have also been proposed to predict the quality (or reliability) level of software products [Farr, 1996]. However, practitioners in the software industry continue to face problems related to cost overruns.

Prior research has not been able to provide answers since productivity and quality modeling efforts have often considered either only the productivity or the quality. That is, most productivity models ignore the quality of the delivered product, and quality models ignore the cost incurred in developing or maintaining the products. A key reason is that the accounting systems of software organizations lack provisions for tracking life-cycle costs. Moreover, the effect of development process aspects has not been explicitly incorporated in cost or quality models. Empirical evidence on the effect of process factors is restricted to case studies and experience reports of a few projects. Our field study fills this void.

The three main contributions of our field study are: 1) We develop models for software life-cycle productivity that include both development and maintenance costs. Most previous models address either development or maintenance costs separately. 2) Our models also capture the effect of

the software development process on life-cycle productivity and quality. 3) We empirically validate these models using primary data on projects of a large commercial software developer.

3. Theoretical Framework

As noted earlier, software managers and executives face a multitude of choices that impact productivity and quality. The major choices relate to technology, people, process, and product factors. However, the specific impacts of these factors on life-cycle productivity and quality are still not clear. For instance, should a manager invest in the latest software development approach and language, or in an automated tool to support software design? Likewise, should the manager hire new programmers or invest in process improvements? At present, the lack of a rigorous framework can lead to incorrect trade-off.

In our framework, we address the following research questions:

1. What is the trade-off between quality and life-cycle productivity?
2. What are the effects of the development process on life-cycle productivity and quality?
3. Does up-front resource deployment pay off?
4. What are the effects of development resources on productivity and quality?

We examine each of these questions in the subsections 3.1-3.4.

3.1 Quality and Life-Cycle Costs

We build on the research in manufacturing on the trade-off between quality and productivity. There are many parallels between costs of quality in manufacturing and those in software development. The relationship between productivity and quality in the manufacturing context has been discussed from two viewpoints [Garvin, 1987]. The first perspective, a traditional viewpoint, asserts that increased expenditures are required to attain higher quality levels and highlights economic conformance levels of quality. The second perspective considers the life-cycle cost of the product,

i.e., total cost incurred in product development and service support. The claim made is that costs are inversely related to the quality attained and it is always optimal to produce products with zero defects [Crosby, 1979; Gyrna, 1988]. The rationale for this view is that cost reduction and quality improvement can be simultaneously attained by reducing waste and rework, and that reducing defects in the product leads to substantial savings in support costs. We translate these perspectives to the software domain in our modeling effort.

3.2 Development Process

The methods and practices adopted by software development professionals to develop and maintain software are believed to have significant impact on the project outcomes in terms of development cost and product quality. Disciplined methods and practices such as requirements analysis, defect prevention, and configuration management are expected to result in better control over the software development process. We measure these key process areas using the Capability Maturity Model (CMM) [Paulk et al., 1993a;1993b].

3.3 Resource Deployment

An important managerial question involves the relative deployment of resources across various stages of software development, such as feasibility study, requirements analysis, high-level and low-level design, and coding, etc. It has been observed in the literature that allocating more resources in the early stages may improve the quality and productivity of the software considerably [Blackburn, Scudder, and van Wassenhove, 1996; Humphrey, 1989]. The rationale behind this argument stems from the importance of requirements analysis and design. For instance, when customer requirements are not well mapped into the product design, customers may find more defects in the final product due to induced changes in the later stages of development. As a result, the quality of the end product is likely to suffer.

We examine the impact of varying deployment of resources in the early stages of product development on software conformance quality. All the software products at our research site were developed using the sequential life-cycle model often known as the "waterfall" model. This systems development model specifies distinct stages of product development such as feasibility study, requirements analysis, detailed design, coding and unit testing, systems integration, and field maintenance [Boehm, 1981].

3.4 Development Resources

The effect of the technical capability of the members of a software development team and the usage of various software tools in determining the quality of the product or productivity of the team are well addressed in the literature [Conte et al., 1986; Fenton, 1994]. Researchers have used various measures such as average language experience, software domain experience and analyst experience as proxies for the technical capability of the teams. Field studies in software projects have reported both positive and negative effects of tool deployment on software productivity [Banker et al., 1992]. In order to control for these variations in productivity and quality, we include measures for both personnel capability and usage of tools in our model. Note that the effect of these drivers can be quantified after the software size has been accounted for.

4. Research Model and Data Collection

In this section, we first develop the conceptual elements of our research model. We then describe the research site, and data collection methods to test the model. This section ends with a description of the measures used for the variables in our model.

4.1 Conceptual Model

A schematic diagram of the conceptual elements of our research model is shown in Figure 1. The model addresses the research questions related to tradeoffs between life-cycle productivity and quality and the effects of resource deployment and process design. The primary links of interest in our

study are represented by bold arrows in the figure, and mapped to the research questions in Sections 3.1-3.3. The oval shapes and lighter arrows in Figure 1 show the control variables discussed in Section 3.4.

---------- Insert Figure 1 Here ----------

The life-cycle productivity of a product is defined as the ratio of product size in KLOC by total life-cycle costs. The life-cycle cost includes both development and support costs. Development costs include the costs incurred in all the stages before shipping the product to the field, whereas support costs are incurred in fixing customer reported problems. Once the software product is released to the customers in the field, if the quality of the product is inadequate, customers report a significant number of problems. Hence, for a poor quality product, software vendors may incur substantial support costs to fix the problems reported by the customers.

Thus we summarize the interaction between quality and life-cycle productivity depicted in Figure 1 using the following pair of equations. In Equation 1, we specify quality of the product (**QUALITY**) as a function of product size (**SIZE**), personnel capability (**PER-CAP**), usage of tools (**USG-TOOL**), process factors (**PROCESS**) and proportion of investment in the front-end of development (**FRNT-RES**). Similarly, we specify product life-cycle productivity (**LC-PRODUCTIVITY**) in Equation 2 as a function of quality and factors related to tools, development process and people.

$$\textbf{QUALITY} = f_1(\textbf{PER-CAP, USG-TOOL, SIZE, PROCESS, FRNT-RES}) \quad ...(1)$$

$$\textbf{LC-PRODUCTIVITY} = f_2(\textbf{QUALITY, PER-CAP, USG-TOOL, PROCESS}) \quad ...(2)$$

We describe our research site to test the above model in the following section.

4.2. Research Site and Data Collection

Our research site is one of the largest software development laboratories of a Fortune 100 company. This laboratory develops commercial software systems for various applications. The annual revenue of the firm exceeds several hundred million dollars and the lab employs over 2500 software professionals. In the recent past, various practices for improving quality and productivity have been instituted. Considerable resources are being deployed at the design and planning stages of product development. In addition, efforts have been made to improve the process adopted for developing the software products. Our study was initiated to assess the effectiveness of these programs.

Our data collection involved gathering information on the costs and quality of a cross section of systems software products developed for various markets. We chose recent projects in order to control for the change in productivity or quality of the software projects due to tools and development technology. Based on our discussions with the managers, we started with an initial sample of 56 projects. We dropped thirteen products due to incomplete data. These projects were dropped primarily due to lack of reliability and validity in their metrics. For example, in some of these projects, a major part of the development effort was outsourced to a software contractor. But the company did not track the cost data for the contractor's effort on a project basis. In addition, the quality variables based on defect counts were not accurate in these projects. There was a lack of consistency in tracking the defects especially when it was related to the contractor code. Note that all the projects in our sample are from a single firm. In empirical research on software projects, in order to control for various organizational specific effects and consistency of measures, choice of data samples from a single firm is often preferred [Maxwell, Wassenhove and Dutta, 1999; Banker and Slaughter, 1997; Boehm, 1981].

Our sample size of forty-three is not small compared to other studies of costs or quality in software products. For example, the COCOMO cost model, based on one of the larger datasets, is

estimated with data on sixty-three software projects [Boehm, 1981], whereas Albrecht and Gaffney [1983] used twenty-four projects and Kemerer [1987] used fifteen. Since most software organizations do not have a standard software metrics program to track quality and productivity data in the various stages of product development, it is extremely difficult to find reliable data on software cost and quality for a large sample size. In addition, data on recent projects are often considered proprietary by software organizations.

Our productivity measure includes data on costs (in dollar figures) incurred in all stages of product development and support starting from product planning through service support in the field. Since the software products in our sample were developed over a span of five years, we normalize the cost figures to constant 1993 dollars using an appropriate normalization table [PRICE, 1994]. All the products in our sample were developed in C and a similar proprietary programming language.

4.3 Variable Definitions

The following variables are used in our analysis:

SIZE : The size of each product is measured in terms of lines of code. Although this measure has been widely used in the software engineering literature, certain problems associated with this measure are well known [Kemerer, 1987; Jones, 1991]. The main shortcoming of this measure stems from the inaccurate and inconsistent definition of "a line of code" across various programming languages and tools used to count the number of source lines of code. In order to ensure consistency of measurement across products, the size of each product in our analysis was measured using the same in-house tool. This tool measures the number of executable source instructions, excluding blank and comment statements. This count of executable statements is recognized as a more accurate measure than a count of the physical lines of code [Park, 1992]. Most of the products in our sample were new products and a few were new subsystems to existing products. Hence we used the count of new lines of code for our size measure.

QUALITY (Conformance-Quality): Our quality measure captures the number of unique problems reported by customers. In order to control for the size effects, this measure is normalized by the product size measured in thousand lines of code. Since the product cycle times in this industry are relatively shorter and most of the defects for the products in our study were reported in the first seven quarters after customer shipment, in our quality measure, we use the number of unique problems reported in the first eight quarters after product shipment. For ease of interpretation, we define our quality measure as thousand lines of code per customer reported defect. Thus our quality measure is normalized for product size.

LC-PRODUCTIVITY (Life-Cycle Productivity): We define life-cycle productivity as a ratio of product size to the total cost incurred in product development and support. Our cost data is in dollars instead of person hours to avoid problems related to aggregating the effort of experienced and novice programmers. Our cost data was obtained by computing the work hours from the weekly online time-sheets of the software professionals working on the projects and overhead costs allotted to the projects.

USG-TOOL (Usage of Tools): This variable captures the usage of automated software tools across various stages of product development. We measure this construct using a five point Likert scale available in the literature [Boehm, 1981]. The product manager and at least two randomly selected software engineers from the team rated the usage of tools on this five-point scale. The final score for each product was obtained by averaging these responses. The inter-rater reliability index of 0.69 for this measure was also well above the threshold recommended by Nunnally [1967].

PER-CAP (Personnel Capability): This variable measures the technical capability of the members of the product team. In our analysis, we measure this construct using a five point Likert scale available in the literature [Boehm, 1981]. The product manager and at least two randomly selected software engineers from the team rated the capability of the team on this five-point scale.

The final score for each product was obtained by averaging these responses. The inter-rater reliability index for this measure was 0.72.

FRNT-RES (Front-End Resources): This variable measures the percentage of the development resources (i.e., total cost incurred before shipping the product to the customers) deployed in the initial stages of the development process, including product planning, high-level design and detailed design. As noted earlier, the projects in our sample were either new development or addition of new subsystems with several thousand lines of code. Hence this measure of front-end investments is not biased by minor modification projects.

4.4 Process Measurements

As discussed earlier, in our analysis, we consider the software process areas specified in the CMM that are relevant to software productivity and quality. The CMM is widely known in the software industry and it highlights 18 process areas in the software development process [Paulk, et al., 1993b]. The CMM also identifies specific practices to be followed in each process area. Software managers adopt these practices in their development process in order to improve the quality and productivity of their projects. In our study, we assessed all the projects based on the degree to which these practices were consistently adopted across projects. Although all the projects in our data were from a single organization, our measurements of CMM process areas at the project level exhibit considerable variance. This is because product managers at our research site had high level of autonomy in resource deployment and process related decisions in their respective projects. Project managers differed in the extent to which they implemented the key practices specified in the CMM key process areas. In addition, the measurement scales that we use in our analysis are adapted for project level data. Our process measurement scale and choice of process areas are discussed in the Appendix. Further details on our process measurements at the project level can be found in Krishnan and Kellner [1999].

Based on the key process areas specified in the CMM model, we define process constructs **Q-PROCESS** and **LC-PROCESS** related to quality and life-cycle productivity of software products respectively. The details of our measurement of these constructs and the specific CMM process areas included are provided in the Appendix. We hypothesize that projects with high scores on the **LC-PROCESS** and **Q-PROCESS** constructs will exhibit higher productivity and quality respectively. The summary of our data is presented in Table 1.

---------- Insert Table 1 Here ----------

5. Data Analysis

In this section we describe the key data analysis procedures in four sub-sections.

5.1 Linear versus Non-linear specification

We consider alternate specifications (linear versus non-linear) for the relationship between various explanatory variables and quality or life-cycle productivity. We contend that this relationship is inherently non-linear. For example, though higher investment in the early stages of product development may improve quality, the improvement in quality may not be linear. Similar arguments may apply for the effect of personnel capability, usage of tools or product size on productivity or quality of software projects. Hence many empirical models on software productivity and quality adopt multiplicative specifications (Boehm [1981]; Maxwell, Dutta and Wassenhove [1999]). Based on the theoretical models described in Section 4, we estimate the parameters of the following multiplicative model specifications to understand the relationship between life-cycle productivity and quality.

$$ln(\textbf{QUALITY}) = \alpha_0 + \alpha_1 * ln(\textbf{SIZE}) + \alpha_2 * ln(\textbf{PER-CAP}) + \alpha_3 * ln(\textbf{USG-TOOL}) + \alpha_4 * ln(\textbf{FRNT-RES}) + \alpha_5 * ln(\textbf{Q-PROCESS}) + \varepsilon_1 \quad \ldots.(3)$$

$$ln(\textbf{LC-PRODUCTIVITY}) = \beta_0 + \beta_1 * ln(\textbf{QUALITY}) + \beta_2 * ln(\textbf{PER-CAP}) + \beta_3 * ln(\textbf{USG-TOOL}) + \beta_4 * ln(\textbf{LC-PROCESS}) + \varepsilon_2 \quad \ldots.. (4)$$

The specification tests for non-nested models rejected linear models over the log linear specifications mentioned above [Davidson and MacKinnon, 1981].

5.2 Estimation Procedures

We estimated the parameters of the two equations in (3) and (4) using OLS (Ordinary Least Squares) estimators. The OLS estimates of the parameters of the two equations are provided in third columns of Table 2A and 2B respectively. Since all the projects are from the same firm, there is a possibility of correlation between the error terms in the two equations. Thus we also treated the two equations as SUR (seemingly unrelated regressions), and estimated the SUR parameters using the generalized least squares estimator allowing for correlation in the error terms across the two equations. The SUR estimates of the parameters of the two equations are presented in fourth columns of Table 2A and 2B respectively. In the SUR model, we tested the significance of the model using a F (J, MT-K) test for linear restrictions (Greene, 1993 pg. 491). In this stastistic M is the number of equations, T is the number of observations per equation, K is the total number of parameters that are estimated and J is the number of linear restrictions in the hypothesis. In our hypothesis, we restricted the coefficients of all the explanatory variables in the two equations to zero and hence J = 9. The computed statistic $F_{9,75} = 27.76$ rejects the null hypothesis at 1% level of significance. Note that the SUR estimates are nearly the same as OLS estimates. This indicates that there is no significant effect of simultaneous correlation between the error terms in the two equations.

5.3 Testing for Endogeneity

It may be argued that quality and productivity are endogenous, i.e., the two variables are determined simultaneously. In other words, it can be claimed that quality affects productivity and productivity affects quality. We tested for the endogeneity of quality variable in the productivity equation using Hausman's endogeneity test (Hausman [1978]). In this test, we estimate the predicted values of the quality variable by regressing the quality variable on all the exogenous variables, i.e., *ln* (**SIZE**), *ln*(**PER-CAP**), *ln*(**USG-TOOL**), *ln*(**FRNT-RES**), *ln*(**Q-PROCESS**) and *ln*(**LC-PROCESS**). This predicted value of the quality variable is then added as an independent variable to

the productivity model along with the original quality variable and other independent variables and model parameters are estimated using OLS. A test for significance of the coefficient of this predicted quality variable indicates if the quality variable is endogenous. In our analysis we did not find this coefficient to be significant. Hence endogeneity of the quality variable in the productivity equation is not a serious concern.

Since the small sample properties of the endogeneity tests are not certain, in order to correct for any bias in the OLS estimates due to potential endogeneity of quality variable in the productivity model, we also treated the two equations (3) and (4) as simultaneous equations and estimated the parameters using 2SLS (two stage least squares). We used the variables *ln* (**SIZE**), *ln*(**PER-CAP**), *ln*(**USG-TOOL**), *ln*(**FRNT-RES**), *ln*(**Q-PROCESS**) and *ln*(**LC-PROCESS**) as the instruments in the first stage of 2SLS estimation. The 2SLS estimates of the parameters of the two equations are provided in fifth columns of Table 2A and 2B respectively. As shown in the tables, the 2SLS estimates are very similar in sign, significance and magnitude to the OLS estimates. This indicates that the OLS estimates are not biased due to endogeneity problems. We also conducted a different version of the Hausman's specification error test for endogeneity based on the differences in the OLS and 2SLS estimates of the quality coefficient in the productivity equation and found no evidence for endogeneity of the quality variable.

5.4 Other Tests

We tested for standard assumptions for these estimators in our sample. The χ^2 test statistics (0.964 and 0.990 for Equations (3) and (4) respectively) with two degrees of freedom, in the test for the normality assumption of the residuals, did not reject the assumptions of normality of the error terms at 5% level of significance [Bowman and Shenton, 1975; Kmenta, 1986]. The presence of multicollinearity, i.e., significant correlations among the independent variables may also influence OLS estimates. We checked for the effect of multicollinearity using conditions specified in Belsley et

al. [1980]. The condition indices for both the equations were below 20. The variance inflation factors for the independent variables in both equations ranged from 1.01 to 1.92, indicating absence of any serious multicollinearity effects. Another standard assumption for OLS estimator is constant variance in the error terms of all the 43 projects in the regression equation (i.e., homoskedasticity) [Kmenta, 1986]. We tested for homoskedasticity in both the equations using White's test and did not find any violations [White, 1980]. However, note that in our analysis, variables such as personnel capability, usage of tools and software process are measured on a subjective ordinal scale. This may lead to biased estimates due to the potential measurement errors in these variables.

Since our sample included projects with varying sizes that incurred significantly different costs, we tested for the effect of influential observations on our parameter estimates. We deleted each observation in the sample, and re-estimated the parameters and computed the Cook's distance between the old and new values in the parameter space [Cook and Weisberg, 1982]. The maximum values of Cook's distance were 0.16 and 0.55 for the quality and productivity equations respectively. The calculated values of F-statistics for Equations (3) and (4) exceeded the critical values at 1% significance. Note that our models also explain a significant amount of variance in the respective dependent variables.

---------- Insert Table 2 Here ----------

6. Results and Discussion

6.1 Drivers of Quality and Life-Cycle Productivity

Our findings in this field study provide several insights for managers of commercial software products. The results of the quality equation identify several drivers of product quality. For example, we find that size of the software product (α_1= 0.66), personnel capability of project team members (α_2 = 1.08), front-end investments (α_4 = 0.22), and software process (α_5 = 1.07) significantly affect quality. Our findings confirm the importance of personnel capability of the software team members in

determining the quality of the product. We also find that investments in the early stages of software development improve quality. Our results support the theory on the importance of customer requirements analysis and the design process adopted for software projects [Blackburn et. al, 1996a; Krishnan, et. al, 1997; Humphrey, 1989].

The effect of size in determining the quality of the product in our analysis needs to be interpreted with care. Note that our quality variable is the ratio of size in KLOC over the number of defects. That is, the quality measure is the reciprocal of defects normalized for size. However, since the effect of size on quality is often non-linear, we include size as an explanatory variable in the model. A larger software product is likely to have several modules, leading to many possible interactions between the modules. Therefore, the likelihood of a defect in a larger product is increased. However, our result (α_1= 0.66) indicates that this rate of increase in the defects due to increase in product size is decreasing for larger products (Note that $|\alpha_1| < 1$). Although it may be argued that the effect of size may be spurious due to the inclusion of size in defining the quality variable, this is not a serious concern in our multiplicative model specification. We estimated the model without normalizing the dependent variable quality for size (i.e., using dependent variable as 1/defects). We find that all the coefficients in the model except for the size variable remains the same. The coefficient of size becomes -0.34 (i.e. α_1= 0.66 - 1 = - 0.34) and was significant. This is due to the removal of linear effect of size from the dependent variable. This indicates the direct negative association of size and quality (not normalized for size). That is, the direct positive association between size and number of defects.

Our analysis highlights the significance of software development process factors in determining the quality of a software product. We find that adherence to the practices of certain development process areas as specified in the CMM framework is associated with higher conformance quality in software products. These practices work in two ways. First, practices such as

code inspections, structured code walkthroughs and causal analysis of the defects as specified in the key process areas of defect prevention and peer review aid in detecting the software problems earlier, thus avoiding the injection of defects. Second, practices such as promoting process visibility within the group, consistently performing quality assurance audits and checks for assessing integrity of various work products provide better control over the software work products, thus reducing the number of defects shipped to the customers. As discussed earlier, the effect of software development process factors on productivity or quality has not been empirically tested before. Our study is one of the first empirical validations of software development process effects on productivity and quality. It is interesting to note that we do not find any significant effect of the **LC-PROCESS** variable on productivity (β_4 is not significant). One explanation for this could be that some of the practices in the CMM key process areas such as software project planning and requirements and training programs may also add to the project overheads in a single project and thus negatively affect productivity. As a consequence the net effect of this process construct is not significant.

In our analysis, we do not find the usage of tools playing a significant role in explaining the quality or productivity. We discussed these results with the managers at our research site and learned that the usage of tools had mixed effects on the software projects. In some projects, version control and documentation tools were efficiently used and provided better control on the software management, thus increasing productivity and quality. In other projects, some design and reengineering tools led to higher learning costs and quality problems due to code mismatch with the tools. It should be noted that our measurement construct on the usage of tools is on a subjective scale. Thus the lack of significant effect of tools in our study may also be attributed to measurement errors. The effect of tool usage in software products needs to be further investigated on a larger dataset measuring the effect of similar types of tools.

Our estimation of the life-cycle productivity in Equation (4) indicates significant increases in life-cycle productivity for improving the quality of the product ($\beta_1 = 0.56$). Note that since $|\beta_1| < 1$, the increase in life-cycle productivity from improving the quality is at a decreasing rate. That is, the marginal payoff in terms of increases in productivity diminishes at higher levels of quality. As noted earlier, life-cycle productivity in our model aggregates effort in development and maintenance phases. We further investigated the effect of **QUALITY** on development and maintenance productivity separately. Our results based on these models indicate that improving quality leads to increases in both development and maintenance productivity. However, from a return on investment perspective, software managers are interested in the total costs, i.e., life-cycle costs incurred in software projects. Our findings in these models also provide evidence for significant positive effect of personnel capability on maintenance productivity. We next discuss the returns from quality improvements on life-cycle productivity in the following subsection.

7.2 Quality Improvement and Returns

Given that multiple options for investing in the product are always open for managers, a quantitative assessment of the payoff from hiring the best people, improving the process, and ensuring conformance quality in the products is important. In our analysis, we find that the personnel capability of the team members has a direct, significant effect in improving quality. Thus our results show evidence for both direct and indirect (through quality improvement) effects of personnel capability on software productivity. Our findings also provide evidence for the positive effect of certain software development process aspects in improving the quality of the products.

Our model allows us to quantify the marginal gains of quality improvement on life-cycle productivity. The life-cycle productivity model in Equation (4) is linear in logrithmic transformation of the variables. In this model, holding the variables **PER-CAP**, **USG-TOOL**, and **LC-PROCESS**

constant at their mean levels, the marginal change in life-cycle productivity for improvement in quality is given in Equation (5) below.

$$\frac{\partial(\text{LC}-\text{PRODUCTIVITY})}{\partial(\text{QUALITY})} = \beta_1 * (\text{LC-PRODUCTIVITY})/(\text{QUALITY}). \qquad (5)$$

Our results help to translate quality-related problems into dollar value and thus can help managers to cost justify certain quality improvement programs such as enhancing the development process and hiring the best people. In Figure 2, we show the change in life-cycle productivity (size per $1000) for 1% improvement in quality at various levels of quality. At higher levels of quality, the payoff in life-cycle productivity decreases. It can be argued that there exists a threshold quality level such that any improvement in quality beyond this level may lead to a decrease in life-cycle productivity. Thus quality is not free beyond this threshold. However, in the current range of quality levels in our analysis, we do not find evidence for such a threshold level of quality.

---------- Insert Figure 2 Here ----------

Figure 2 also provides managerial insights for resource allocation decisions. For example, beyond a certain level of conformance quality, managers may find it beneficial to allocate resources to other dimensions of product quality, such as functionality or performance of the products. Note that our marginal analysis is based on the mean value of the OLS estimate of the parameter β_1 and does not include the variance in OLS estimate of β_1. For example, allowing for single standard deviation error around the mean estimate of β_1, our analysis indicates that at the mean level of quality, the productivity gains from 1% improvement in quality may rage from 0.46% to 0.66%.

7. Conclusion

Our field study of commercial software production in a leading software company provides several insights for product managers. First, our results provide evidence for significant increases in life-cycle productivity from improved quality in software products shipped to the customers. Given

that the expenditure on computer software has been growing over the last few decades, empirical evidence for cost savings through quality improvement is a significant contribution to the literature. Second, our study identifies several quality drivers in software products. Our findings indicate that personnel capability, software development process factors, and deployment of resources in the initial stages of product development (especially design) have significant positive impact on the quality of the product. We find that although larger products exhibit higher number of defects, the rate of increase in the defects due to increase in product size is decreasing for larger products. We find no evidence in support of the effect of the usage of tools on quality or productivity. Hence, our analysis provides insights to help product managers improve product quality and life-cycle productivity.

We also find both direct and indirect effects from the personnel capability of software team members on the maintenance productivity. Software firms have deployed significant resources to adopt the CMM and other similar models for software process improvements. However, as discussed earlier, the effect of process on the outcomes of the project was not tested empirically prior to our field study. Our findings indicate positive association between product quality and certain development practices specified in the CMM model.

As discussed in Kriebel [1979], quality of a software product is multidimensional. Our study primarily addresses conformance quality in software products. It is recognized that building more features into the product or supporting a software product in multiple environments may only increase the costs to the software vendor. Further research is required to define objective metrics for other dimensions of software quality and to study their effects on productivity. It is possible that in addition to project size, other project complexity measures such as code complexity or number of modules and their interactions may also determine product quality. However in our sample, we do not have reliable data on these measures of complexity. Another limitation of our analysis is that variables such as personnel capability, software process and usage of tools are measured on a subjective ordinal scale.

However, since these variables are not dependent variables in our models, the OLS estimates of the model parameters are not affected. Although we treat the final score on these variables as continuous variables, coefficient estimates of these variables should be interpreted with caution. Future studies should attempt to define objective metrics to measure personnel capability and tools and measure the adoption of specific practices specified in the CMM key process areas and link these practices to productivity and quality.

Our analysis is restricted to data collected from a single organization. The limited variance in the data from a single organization could also be a reason for the lack of significance of the effect of variables such as usage of tools on productivity and quality. However, pooling data from multiple firms would need control for accounting standards for cost data and consistent measurement of product size, cost and quality. In addition, organizational structure of the firms and other environmental variables may affect the economics of quality. Though obtaining clean data from multiple organizations with appropriate measurement controls is difficult, research that pools data from multiple sites with adequate controls for various factors discussed above is needed to further validate our results.

Our findings on the marginal productivity gains from quality improvements should be interpreted with caution. The range of conformance quality in the software products differs depending on the software domain. For example, the quality range for commercial systems software products (as at our research site) may be significantly lower than the quality range in mission-critical software (such as the NASA space shuttle or medical diagnostic systems). Further research is required to assess the change in life-cycle productivity at the various levels of quality of software in these domains. Software organizations may need to estimate the shift of the curve depicted in Figure 2 for their respective software environments.

Appendix

Software Process Constructs

The Capability Maturity Model (CMM) of the Software Engineering Institute is widely known in the software industry and it highlights 18 process areas in the software development process [Paulk, et al., 1993b]. The CMM also identifies specific practices to be followed in each process area. Some of the CMM process areas address process aspects at an organizational level, while others address software practices at the project level. Since the unit of our analysis is software projects within an organization, we consider only those process areas which address project level software practices. Our list of process areas is shown in column one of Table A1.

---------- Insert Table A1 Here ----------

In all the 18 process areas, the CMM specifies two to four questions on software practices related to the goals of each key process area. We developed an instrument using the questions related to these practices on a five point scale with anchor values from 1 ("The practices were never performed ") to 5 ("The practices were performed more than 90% of the time"). We used exactly the same questions for each process area, as specified in the CMM [Paulk, et. al., 1993b]. This scale measures the degree to which these practices were followed in the respective projects. We checked for the content validity of the instrument through pilot testing with the members of the Software Engineering Institute and other software professionals[1]. We also verified the construct validity of the instrument based on the approach described by Benbasat and Moore [1991]. The instrument was suitably modified after pilot testing at our research site. A detailed discussion on our measurement scale, content and construct validity can be found in Krishnan and Kellner [1999].

[1] We thank Prof. Barry Boehm of the University of Southern California, Dr. Dennis Goldenson and Dr. Dave Zubrow of the Software Engineering Institute, for their useful comments in the construction of our instrument.

In order to check for reliability, we obtained multiple responses for each software product. Our respondents were product managers and at least two randomly selected software engineers from the product team. The inter-rater reliability of these measures for the same software project was also above the threshold recommended by Nunnally [1967]. A score on each question was obtained by averaging these responses. Similarly, a score on each process area was obtained by averaging the scores on all the questions in that process area. All the process areas exhibited a high reliability index as measured by Cronbach alpha (minimum of .87 for project planning to maximum of .95 for requirements management). In addition, confirmatory factor analysis of the responses to various questions in each process area identified that one factor in each process area explained over 70% of the variance. We also verified that all questions loaded almost equally on one factor and only one factor was retained on minimum eigen value criterion in each process area. The factor loading and reliability measures for the process areas are provided in Table A1. For ease of interpretation, we weigh all the practice questions within a process area equally for computing the final score of each process area.

Broadman and Johnson [1994] report results on the perceived effect of the CMM process areas on the cost and quality outcomes in software projects. Based on their results and discussions with the original architects of the CMM model, we develop software process constructs **Q-PROCESS** (quality related process areas) and **LC-PROCESS** (cost related process areas) in our quality and productivity specifications respectively. As shown in Table A2, we measure four process areas in each of these two constructs. For example, configuration management process area in the **Q-PROCESS** variable specifies disciplined practices adopted to manage various configurations of software work products, such as source files, documentation, etc. A proper control of these work products would lead to reduction in software errors due to incorrect versions of source files and documents. Similarly, quality assurance, defect prevention, and peer review process areas address

disciplined practices related to defect prevention and defect detection. The **LC-PROCESS** construct in our model includes requirements management, project planning, training program, and product engineering. These process areas specify disciplined practices to improve productivity of software development and maintenance, and thus life-cycle productivity.

---------- Insert Table A2 Here ----------

The final scores on the variables **Q-PROCESS** and **LC-PROCESS** were obtained by averaging the scores on the respective set of four process areas as depicted in Table A2. We also verified through confirmatory factor analysis that these sets of four process areas loaded on one factor. As shown in Table A2, the four process areas in the **Q-PROCESS** construct loaded on a single factor (explaining 69% of variance), and this variable also exhibited a high reliability index as measured by the Cronbach alpha (.840). Similarly, the four process areas in the **LC-PROCESS** construct loaded on a single factor (explaining 70 % of variance) and this variable also exhibited a high reliability index as measured by the Cronbach alpha (.844).

References

Albrecht, A., and J. Gaffney, "Software Function, Source Lines of code, and Development Effort Prediction: A Software Science Validation," **IEEE Transactions on Software Engineering**, Vol.SE-9, No.6, pp 639-647, November, 1983.

Banker, R. D., R. J. Kauffman, R. Kumar, "An empirical test of object based output measurement metrics in Computer Aided Software Engineering (CASE) Environment," **Journal Of Management Information Systems**, Vol 8, No 3, 1992, pp 127-150.

Banker, R.D., S. Datar, and C.F.Kemereer, "A model to evaluate variables involving the productivity of software maintenance projects," **Management Science**, Vol 37:1, January 1991, pp 1-18.

Banker, R.D., S. Datar, C. F. Kemerer, and D. Zweig, "Software complexity and software maintenance costs," **Communication of the ACM**, vol 36, pp. 81-93, November, 1993.

Banker, R. D. and S. Slaughter, "A Field Study of Scale Economies in Software Maintenance," **Management Science**, Vol 43, No 12, pp 1709-1725, 1997.

Belsley, D. A., E. Kuh, and R. E. Welsch, **Regression Diagnostics: Identifying Influential Data and Sources of Collinearity**, New York,: Willey, 1980.

Blackburn, J. D., G. Scudder, L. N. van Wassenhove, "Improving Speed and Productivity of Software Development," INSEAD **Working paper**, February, 1996a, INSEAD Fontainebleau, France.

Blackburn, J. D., G. Hoedemaker, and L. N. van Wassenhove, "Concurrent Software Engineering: Prospects and Pitfalls," **IEEE Transactions on Engineering Management**, vol 43, No 2, May 1996b, pp 179-188.

Benbasat, I. and G. C. Moore, "Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation," **Information Systems Research**, Vol 2, No 3, 1991.

Bowman, K. R., and L. R. Shenton, "Ominibus Contours for Departures from Normality Based on root(b_1) and b_2," **Biometrika**, Vol 62, 1975, pp 243-250.

Boehm, B. W., **Software Engineering Economics**, Prentice-Hall, Inc., New Jersey, 1981.

Broadman, J. G. and D. L. Johnson, "Estimating Process Improvements Returns", **Proceedings of the Risk, Metrics, and Cost estimating workshop,** Software Engineering Institute, Pittsburgh, PA, February, 1994.

Conte, S. D., H. E. Dunsmore and V. Y. Shen, **Software Engineering Metrics and Model**, Reading MA, Benjamin-Cummings, 1986.

Cook, R. D. and S. Weisberg, **Residuals and Influence in Regression**, Chapman & Hall: London, 1982.

Crosby, Philip B. **Quality is Free**. McGraw-Hill, New York, NY, 1979.

Curtis W., Kellner, M. I. and J. Over, "Process Modelling," **Communications of the ACM**,Vol. 35, Sept, 1992, pp 75-90..

Davidson, R. and J. MacKinnon, "Several Tests for Model Specifications in the Presence of Multiple Alternatives," **Econometrica**, 49, 1995, pp 781-793.

Davidson, R. and J. MacKinnon, "Several Tests for Model Specifications in the Presence of Alternative Hypotheses," **Econometrica**, 49, 1981, pp 781-793.

Demarco, T., **Why does software cost so much and other puzzles of the Information Age**, Dorset House Publishing, NewYork NY, 1995

Farr, W., "Software Reliability Modeling Survey," M. R. Lyu (eds) **Handbook of Software Reliability Engineering**, McGraw-Hill, 1996, pp 71-115.

Fenton, N. E., **Software Metrics: A Rigorous Approach**, Chapman & Hall, New York, 1994.

Garvin, David A., "Competing on the Eight Dimensions of Quality," **Harvard Business Review**, 101-117, Nov-Dec 1987.

Greene, W. H., **Econometric Analysis**, Macmillan Publishing Company, New York, 1993

Gyrna, F., "Quality Costs," in **Quality Control Handbook**, 4th edition, McGraw-Hill, New York, 1988.

Humphrey, W. S., **Managing the Software Process**, Addison-Wesely, Reading Mass. 1989.

Humphrey, W. S., "Introduction to software process improvement," **Technical Report**, CMU/SEI-TR-92-7, 1992, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

Jones, C., **Applied Software Measurement: Assuring Productivity and Qaulity**, McGraw-Hill Book Co., New York, 1991.

Keil, M., "Pulling the plug: Software project management and the problem of project escalation," **MIS Quarterly**, v19n4, Dec 1995, pp. 421-447.

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

Kemerer, C. F., "Software complexity and software maintenance: A survey of empirical research," **Annals of Software Engineering**, vol 1, pp 1-20, 1995.

Kmenta, Jan., **Elements of Econometrics**, Macmillan, New York, 1986.

Kriebel, C.H., "Evaluating the Quality of nformation Systems", **Proceedings of the BIFOA Symposium**, Sept 18-20, 1979, Bensberg/Colonge.

Krishnan, M. S., M. Kellner, C. H. Kriebel, T. Mukhopadhyay and K. Srinivasan, "Cost, Quality and Customer Satisfaction of Software Products," forthcoming in **Quality Management Series**, Uday Karmarkar and Phill Lederer (eds), Kluver Publications, New Jersey, Forthcoming, 1997.

Krishnan, M. S. and M. I. Kellner, "Measuring Process Consistency: Implications for Reducing Software Defects" Forthcoming in **IEEE Transactions on Software Engineering,** 1999.

Maxwell, K., Wassenhove, Luk. V., and S. Dutta, "Performance Evaluation of General and Company Specific Models in Software Development Effort Estimation," forthcoming **Management Science**, 1999.

Mukhopadhyay, T. and S. Kekre, "Software Effort Models for Early Estimation of Process Control Applications", **IEEE Transactions on Software Engineering**, Vol. 18(10), pp 915-924, 1992.

Nunnally, J., **Psychometric Theory**, McGraw-Hill, 1967.

Park, R. E., "Software size measurement: a framework for computing source statements," **Technical Report** SEI-92-TR-20, 1992, Software Engineering Institute, Carnegie Mellon University, Pitttsburgh, PA 15213.

Paulk, M.C. (1993a), Curtis, Bill, Chrissis, M.B., Weber, C.V. "Capability Maturity Model, Version 1.1", **IEEE Software**, pp 18-27, July 1993a.

Paulk, M. C. (1993b), C. V. Weber, S. M. Garcia, M. Chrissis, and M. Bush, "Key Practices of the Capability Maturity Model, Version 1.1," **Technical Report**, CMU/SEI-93-TR-25, 1993b, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

PRICE **Newsletter**, Vol 18, Number 1, "Today's information source for tomorrow's international computer aided parametric estimating," June, 1994.

Shapiro, S. S. and M. B. Wilk, "An Analysis of Variance Test for Normality," **Biometrika**, vol 52, 1965, pp. 591-612

White, H., "Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity," **Econometrica**, 48, May 1980, pp. 817-838

Figure 1: Conceptual Framework

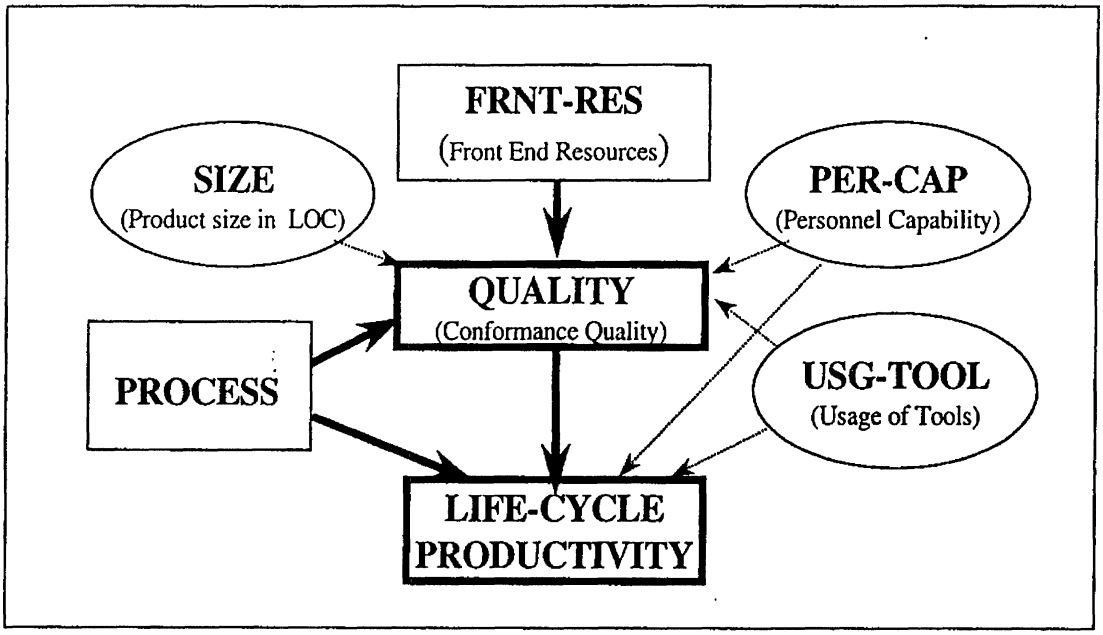

Figure 2: Marginal Increase in Life-cycle Productivity for 1% Improvement in Quality

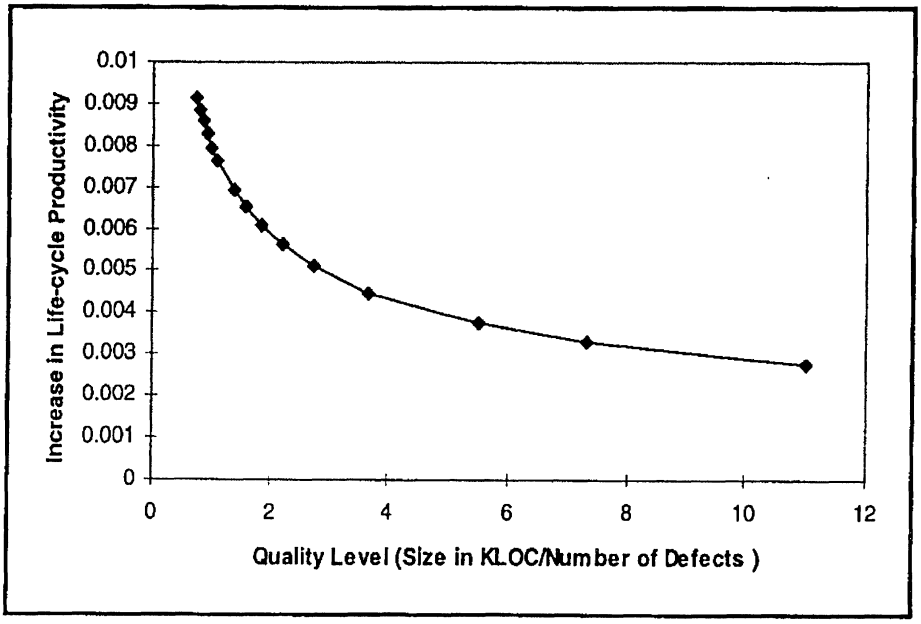

Table 1: Summary Statistics

Variable	Mean	Median	Std Deviation
SIZE (000 Lines of code)	**109.12**	**73.0**	**109.58**
LC-COST (Life-cycle cost in $000)	**13392.35**	**8100.6**	**12284.12**
Number of Field Problems	**80.60**	**66.00**	**56.99**
PER-CAP (Personnel Capability)	**3.46**	**3.5**	**0.775**
USG-TOOL (Usage of Tools)	**2.92**	**3.00**	**0.76**
FRNT-RES (Front-End Investment)	**13.34**	**12.82**	**9.04**
LC-PROCESS (Process in Productivity Equation	**3.31**	**3.27**	**0.73**
Q-PROCESS (Process in Quality Equation)	**3.25**	**3.28**	**0.76**
LC-PRODUCTIVITY (SIZE/LC-COST)	**.010**	**.008**	**.008**
QUALITY (SIZE/Number of Field Problems)	**1.77**	**1.12**	**1.95**

Table 2A: Equation 3 Parameter Estimates
(The t-statistics are given in parenthesis)

Variable Name	Parameter	OLS Estimates	SUR Estimates	2SLS Estimates
Intercept	α_0 (t-statistics)	-5.15** (-5.01)	-5.10** (-4.97)	-5.15** (-5.01)
ln(SIZE)	α_1 (t-statistics)	0.66** (7.01)	0.67** (7.12)	0.66** (7.01)
ln(PER-CAP)	α_2 (t-statistics)	1.08** (2.54)	1.07** (2.52)	1.08** (2.54)
ln(USG-TOOL)	α_3 (t-statistics)	-0.55 (-1.33)	-0.53 (-1.29)	-0.55 (-1.34)
ln(FRNT-RES)	α_4 (t-statistics)	0.22* (1.90)	0.21* (1.79)	0.22* (1.90)
ln(Q-PROCESS)	α_5 (t-statistics)	1.07* (1.78)	1.01* (1.71)	1.07* (1.78)
R^2(adjusted)		0.557	0.556	0.557
$F_{5,37}$ -Statistics		11.58 (p < .001)		

** 5% significance * 10% significance

Table 2B: Equation 4 Parameter Estimates
(The t-statistics are given in parenthesis)

Variable Name	Parameter	OLS Estimates	SUR Estimates	2SLS Estimates
Intercept	β_0 (t-statistics)	-5.04** (-7.18)	-5.04** (-7.20)	-5.04** (-7.14)
ln(QUALITY)	β_1 (t-statistics)	0.56** (5.45)	0.59** (5.89)	0.62** (4.73)
ln(PER-CAP)	β_2 (t-statistics)	0.65 (1.63)	0.63 (1.60)	0.62 (1.55)
ln(USG-TOOL)	β_3 (t-statistics)	-0.07 (-0.16)	-0.04 (-0.10)	-0.02 (-0.05)
ln(LC-PROCESS)	β_4 (t-statistics)	-0.55 (-1.11)	-0.57 (-1.14)	-0.58 (-1.15)
R^2(adjusted)		0.454	0.451	0.44
$F_{4,38}$ -Statistics		9.74 (p < .001)		

** 5% significance * 10% significance

Table A1

Reliability coefficient and Reliability Coefficients

Software Development Process Area	Number of items	Cronbach Alpha
Software Configuration Management	4	.89
Defect Prevention	3	.92
Peer Review	2	.92
Software Quality Assurance	4	.89
Requirements Management	2	.95
Software Project Planning	3	.87
Software Product Engineering	2	.91
Training Program	3	.90

Table A2

Reliability coefficient and factor loadings for process variables

Process Variable	Number of items	Cronbach alpha	Loadings on Factor retained
LC-PROCESS	4	.844	
Requirements Management			.879
Software Project Planning			.809
Software Product Engineering			.857
Training Program			.756
Q-PROCESS	4	.840	
Software Configuration Management			.851
Defect Prevention			.851
Peer Review			.758
Software Quality Assurance			.833