THE EFFECTS OF SOFTWARE PROCESS MATURITY ON SOFTWARE DEVELOPMENT EFFORT

by

Bradford K. Clark

A Dissertation Presented to the

FACULTY OF THE GRADUATE SCHOOL

UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the

Requirements for the Degree

DOCTOR OF PHILOSOPHY

(Computer Science)

August 1997

Version 1.0

Copyright Notification

DEDICATION

This work is dedicated to Deborah Clark. It was through your encouragement, support, and sacrifice that I was able to attend Graduate School. I will be in your debt always. God bless you.

ACKNOWLEDGMENTS

I would like to thank some of the key people from the eighteen organizations that made data available for this research: Jarius Hihn from NASA JPL, Gary Thomas from Raytheon/E-Systems, Charity Nosse from EDS, Don Firesmith from NASA SEL, Stuart Glickman from Bellcore, Sherry Stukes from MCR, and Hillel Myers with others from Motorola. Without their efforts I would not have been able to do the analysis which led to this report.

I would like to acknowledge the support of employer supervisor, Mr. Robert Page. He supported my request for a leave of absence and my request for employer education assistance for graduate school. Since returning to my employer, his continued support made possible the productive use of my education in my everyday work.

I would like to thank Dr. Barry Boehm for being an excellent mentor. He was patient with my learning and always enthusiastic about my research. He helped me when I asked for it. His kindness and generosity made the strain of graduate school bearable. His brilliance motivates me still to keep digging.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

A software product is often behind schedule, over budget, non-conforming to requirements and of poor quality. Controlling and improving the processes used to develop software has been proposed as a primary remedy to these problems. The Software Engineering Institute at Carnegie Mellon University has published the Software Capability Maturity Model (SW-CMM) for use as a set of criteria to evaluate an organization's Process Maturity. The model is also used as a roadmap to improve a software development process's maturity. The premise of the SW-CMM is that mature development processes deliver products on time, within budget, within requirements, and of high quality.

This research examines the effects of Software Process Maturity, using the SW-CMM, on software development effort. Effort is the primary determinant of software development cost and schedule. The technical challenge in this research is determining how much change in effort is due solely to changing Process Maturity when this change generally occurs concurrently with changes to other factors that also influence software development effort.

The six mathematical models used in this research support the following conclusion: For the one hundred twelve projects in this sample, Software Process Maturity was a significant factor (95% confidence level) affecting software development effort. After normalizing for the effects of other effort influences, a one-increment change in the rating of Process Maturity resulted in a 15% to 21% reduction in effort. The modeling approach used in this analysis can be used in other areas of Software Engineering as well.

Chapter 1

INTRODUCTION

There are many companies and government organizations that develop or maintain software to support their operations or their business products. The development of software includes the creation of specification, design, source code, and testing. These different artifacts interact with each other where a delay or defect in one affects the completeness of the others. This often results in a software product that is behind schedule, over budget, non-conforming to requirements and of poor quality. The result is that the company loses money or the government organization misuses taxpayers' money either through budget overruns or decreased user and customer satisfaction. Controlling and improving the process used to develop software is seen as the remedy to these problems [Humphrey 1989].

The Software Engineering Institute has published a Software Capability Maturity Model (SW-CMM) that can be used to rate an organization's software process maturity [Paulk et al. 1995a]. The motivation behind the SW-CMM is that a mature software development process will deliver the product on time, within budget, within requirements, and of high quality. The model is based on five levels; organizations with ad hoc processes start at Level One. To progress to the next higher level, Level Two, an organization has to demonstrate a repeatable process. To gain a Level Three rating an organization has to demonstrate a defined process. A Level Four organization has a managed process and a Level Five organization has an optimizing software development process. The SW-CMM is explained in Chapter 2.

An important question for industry and government is what are the benefits of investing resources to improve the Organization's Process Maturity. The long-term benefits of high process maturity are software delivered on time, within budget, within customer requirements, and of high quality. An important benefit would be the effect it has on productivity. Two experts have expressed significant disagreements [Springsteen et al. 1992]. Using the database associated with his Checkpoint model, Capers Jones predicted that as an organization increases maturity levels its productivity increases. However the burden of oversight groups, compliance checking, and upper management involvement will start causing productivity to decrease at the mid-maturity level. His analysis was focused at the process level and does not consider organization level processes which occur at the higher maturity levels. Another prediction by Larry Putnam, using his Productivity Analysis Database, was that as the Maturity Level increases productivity increases. This is based on the database Productivity Index which is derived from size, effort, and time used to develop the software. It did not separate out other factors that influence productivity, e.g., product complexity, personnel, software technologies, or development practices. It may not be correct to assume a relation exists between Maturity Level and Productivity.

Much has been written discussing the short-term and long-term benefits of increasing maturity levels [Broadman and Johnson 1995, Butler 1995, Dion 1993, Herbsleb et. al. 1997, Humphrey et. al. 1991, McGibbon 1996, Springsteen et. al. 1992, Wohlwend and Rosenbaum 1994]. It requires a large amount of dollar investment by an organization to change the software development process within the organization and to realize an increased level of maturity. The effects of increasing process maturity alone are not easy to determine, as organizations are generally making concurrent improvements in other areas that result in benefits to the development organization.

The reported results would be more convincing if the method of evaluation were able to separate out the effects of other software development factors in addition to Process Maturity. Only then can a concise conclusion be drawn about Process Maturity's effect on productivity. The purpose of this research is to perform a more sophisticated analysis of the effect that increasing Process Maturity has on software development effort, a component of productivity.

The technical challenge in this research is determining the effect that increasing Process Maturity has on effort within the context of other factors that influence software development effort. This involves the collection of data that is based on observations and not on controlled experiments. A mathematical model is proposed that segregates Process Maturity's influence on effort from other influencing factors. The model is analyzed for goodness of fit and accuracy.

The contribution of this research is the discovery of the quantified effect that Process Maturity has on software development effort and the modeling approach used to isolate the effects of Process Maturity on effort. Understanding Process Maturity's influence on effort within the context of other factors provides a trade-off analysis capability that can be used to lower the effort required to produce a software product. The modeling approach can be used in other areas of Software Engineering.

The next Chapter discusses background material: the Software Capability Maturity Model, effort estimation modeling methods, and multiple regression analysis. Chapter 3 reviews the literature on Software Process Improvement using the SW-CMM and on process related inputs to currently available effort estimation models. Chapter 4 discusses the research question drawn from the literature review and presents the modeling approach. Chapter 5 presents the results of this research. Chapter 6 discusses the research conclusions, contributions and future directions.

Chapter 2

REVIEW OF THE SOFTWARE CAPABILITY MATURITY MODEL AND STATISTICAL MODELING OF EFFORT EXPENDITURE

2.1 Software Capability Maturity Model

The Software Capability Maturity Model (SW-CMM) provides a set of requirements that organizations can use in setting up the software process used to control software product development. The SW-CMM specifies "what" should be in the software process but not "when" or "for how long." The SW-CMM has what is called a process maturity framework [Paulk et al. 1995a]. There are five levels of process maturity, Level 1 (lowest) to Level 5 (highest). To be rated at a specific level an Organization has to demonstrate capabilities in a number of Key Process Areas (KPA) associated with a specific SW-CMM level, Table 1. The capabilities demonstrated in transitioning from lower levels to higher levels are cumulative. In other words, a Level 3 Organization must demonstrate KPA capabilities from Level 2 and from Level 3.

The Process Maturity framework is presented in Table 1. All Organizations start at Level 1. This is called the Initial level. At this level few processes are defined, and success depends on individual effort. This makes the software process unpredictable because it changes as work progresses. Schedules, budgets, functionality, and product quality are generally unpredictable.

To achieve Level 2 the organization demonstrates capability in 6 KPA's. A Level 2 Organization has basic management processes established to track cost, schedule, and functionality. Problems in meeting commitments are identified when they arise. Software requirements and work products developed to satisfy requirements are baselined and their integrity is controlled. Software project standards are defined and the organization ensures they are faithfully followed. The project works with its subcontractors to establish a strong relationship. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. Level 2 is called the Repeatable level.

A Level 3 Organization has demonstrated capabilities in an additional 7 KPA's. At this level the software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the whole organization. Projects tailor the standard software process to develop their own unique defined software process. A well-defined process includes readiness criteria, inputs, standards and procedures for performing the work, verification mechanisms, outputs, and completion criteria. Level 3 is called the Defined level.

A Level 4 Organization has added 2 more KPA's to its capabilities. At this level detailed measures of the software process and product quality are collected. Projects achieve

Table 1: Process Maturity Framework

SW-CMM Level	Key Process Areas
Level 1	None
Level 2 Repeatable	Requirements Management
	Software Project Planning
	Software Project Tracking and Oversight
	Software Subcontract Management
	Software Quality Assurance
	Software Configuration Management
Level 3 Defined	Organization Process Focus
	Organization Process Definition
	Training Program
	Integrated Software Management
	Software Product Engineering
	Intergroup Coordination
	Peer Reviews
Level 4 Managed	Quantitative Process Management
	Software Quality Management
Level 5 Optimizing	Defect Prevention
	Technology Change Management
	Process Change Management

control over their products and processes by narrowing the variation in their process performance to fall within acceptable quantitative boundaries. Both the process and product are quantitatively understood and controlled. Level 4 is called the Managed level.

At Level 5 an Organization has capabilities in 3 more KPA's and is in a continuous improvement state. Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies. Software project teams analyze defects to determine their causes. Processes are evaluated to prevent known types of defects from recurring, and lessons learned are disseminated to other projects. Level 5 is called the Optimizing level.

2.1.1 SW-CMM Key Process Areas

Each KPA has a set of goals, capabilities, key practices, measurements and verification practices. The goals and key practices are the most interesting of these because they could be used to assess the impact of a KPA on a project development effort, Figure 1. The goals state the scope, boundaries, and intent of a KPA. A key practice describes "what" should happen in that KPA. There are a total of 52 goals and 149 key practices. All of the KPAs are described in [Paulk et al. 1995a].

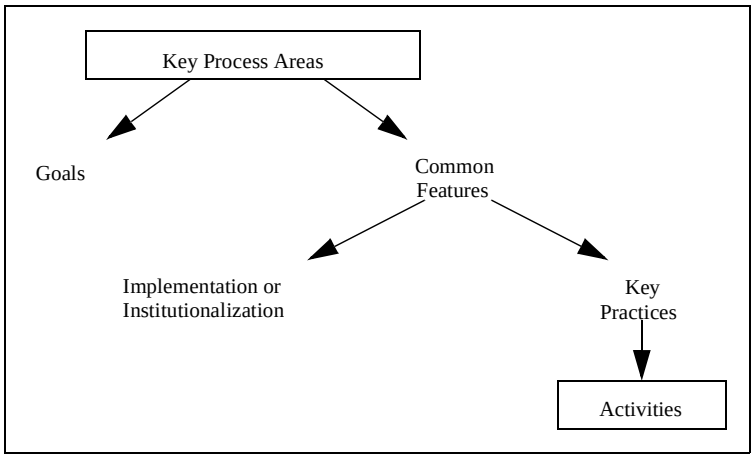


Figure 1. KPA Structure

As an illustration the goals of one KPA from Level 2, Software Project Planning, are given. The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project. Software Project Planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work.

The goals of Software Project Planning are:

1. Software estimates are documented for use in planning and tracking the software project.
2. Software project activities and commitments are planned and documented.
3. Affected groups and individuals agree to their commitments related to the software project.

The top-level activities performed for Software Project Planning are:

1. The software engineering group participates on the project proposal team.
2. Software project planning is initiated in the early stages of, and in parallel with, the overall project planning.
3. The software engineering group participates with other affected groups in the overall project planning throughout the project's life.
4. Software project commitments made to individuals and groups external to the organization are reviewed with senior management according to a documented procedure.

5. A software life cycle with predefined stages of manageable size is identified or defined.
6. The project's software development plan is developed according to a documented procedure.
7. The plan for the software project is documented.
8. Software work products that are needed to establish and maintain control of the software project are identified.
9. Estimates for the size of the software work products (or changes to the size of software work products) are derived according to a documented procedure.
10. Estimates for the software project's effort and costs are derived according to a documented procedure.
11. Estimates for the project's critical computer resources are derived according to a documented procedure.
12. The project's software schedule is derived according to a documented procedure.
13. The software risks associated with the cost, resource, schedule, and technical aspects of the project are identified, assessed, and documented.
14. Plans for the project's software engineering facilities and support tools are prepared.
15. Software planning data are recorded.

2.1.2 Industry SW-CMM Assessments

There are two methods to determine an organization's SW-CMM level, Software Process Assessments and Software Capability Evaluations. The former is done by the organization internally. A team is selected which has been trained in the SW-CMM. The assessment is done with a maturity questionnaire for several projects. The responses are tallied, evaluated and a list of findings are produced. The results become the basis for recommendations for process improvement.

The Software Capability Evaluations focus on identifying risks (such as schedule and budget) on a specific project. The evaluation is performed by the contracting agency during contract bidding. An evaluation team shows up at the contractor's site, interviews are conducted as well as physical evidence of software process artifacts (software requirements documents, policy and procedures documents) are inspected. A list of findings are produced and are used in proposal evaluation.

As of April 1997 the number of Organizations that have had assessments is 542, Figure 2 [Peterson 1997].

2.2 Modeling of Effort Expenditure

There are three approaches used by models to estimate software development effort. Some are based on analogy, some on theory, and others on statistics. The most influential factor in predicting effort in these models is the size of the software product. There are other factors that also affect effort such as product complexity, the application experience of the development team, and development tool support.

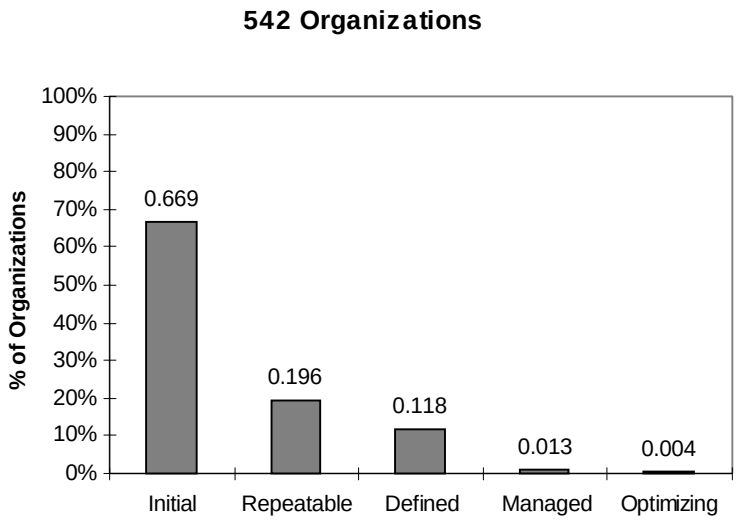


Figure 2. Organization Maturity Profile

2.2.1 Analogy Models

This method of estimating effort is based on the comparison of the planned project with previous projects that have similar characteristics. This model uses experts or stored project data to determine the effort required to develop a software product. For a new product it must be determined what subcomponent level is practical for estimation. There must be an estimate of how many components will likely be in the product. Experts compute the high, low, and most likely estimates for effort required based on the differences between the new and previous projects. It can provide a detailed estimate of effort depending on how deep into the sub-components the analogies are made. The model is weak because the degree of similarity may not be very close to the new project. It is often said that "the devil is in the details."

2.2.2 Theoretical Models

A theory-based estimation model was put forth by [Putnam 1979] and explained in [Conte et. al. 1986, Kitchenham 1990]. It is based on the probability distribution called the Rayliegh curve. This curve express manpower distribution on a project over time, Figure 3. The curve is modeled by the differential equation

$$\frac{dy}{dt} = 2Kat(e)^{-at^2} \qquad \text{Equation 1}$$

where dy/dt is the staff build-up rate, t is the elapsed time from the start of design to product replacement, K is the area under the curve and represents total life-cycle effort (including maintenance), and a is a constant that determines the shape of the curve.

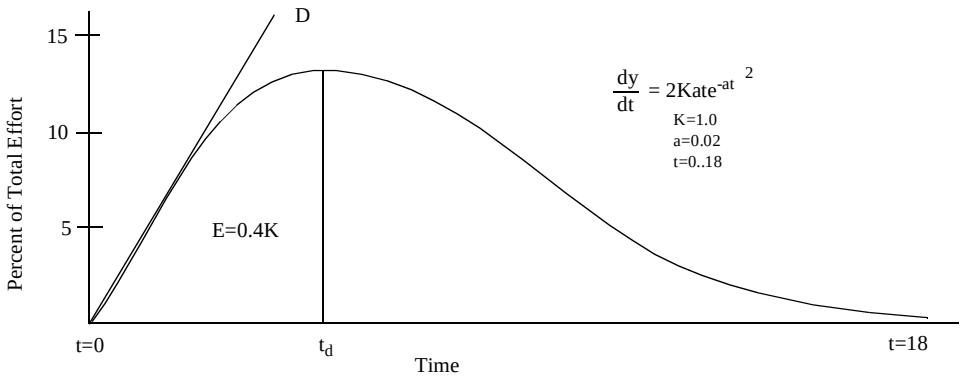


Figure 3. Rayleigh Model

Putnam uses *productivity* to link the basic Rayleigh manpower distribution model to the software development characteristics of size and technology factors. Productivity in software has been defined as the size of the software product, S, divided by the development effort, E: $P = \frac{S}{E}$

To find E in the Rayleigh model, Putnam made the assumption that the peak staffing level (top of the curve) corresponded to the development time. With this assumption, the area under the curve represented development effort, E. E was found to be approximately 40% of K, the total life-cycle effort which is the total area under the curve. Putnam observed from project data that the more productive projects had an initial slower staff buildup and the less productive projects had an initial faster staff buildup. He associated the initial staff buildup of a project with the difficulty of the project, D. D is represented on the Rayleigh curve as the slope of the curve at time t=0. By taking the derivative of Equation 1 and setting t=0, difficulty is defined as:

$$D = \frac{k}{(T_d)^2} \qquad \text{Equation 2}$$

Next Putnam links the Rayleigh manpower distribution and software development effort. He assumes that there must be a relation between difficulty, D, and productivity, P. He finds this relationship to be:

$$P = \alpha D^{-(2/3)} . \qquad \text{Equation 3}$$

Software development productivity is usually defined as the ratio of the software product size to the effort required to develop the product:

$$P = \frac{S}{E} \qquad \text{Equation 4}$$

In Equation 5, Equations 3 and 4 are set equal to each other with D in Equation 3 replaced by its definition in Equation 2 and E in Equation 4 replaced by 0.4K (as explained earlier).

$$\frac{S}{0.4k} = \alpha\left[\frac{K}{t_d^2}\right]^{-(2/3)} \qquad \text{Equation 5}$$

$$S = 0.4\alpha(K)^{1/3}(t_d)^{-(4/3)} \qquad \text{Equation 6}$$

Total life-cycle effort, K, is found to be:

$$K^{1/3} = \frac{S}{0.4\alpha(t_d)^{4/3}} \qquad \text{Equation 7}$$

Equation 8 introduces a *technology factor*, C, which is the product of 0.4 and α. The technology factor accounts for differences among projects such as hardware constraints, personnel experience, and programming environment. Putnam suggests using 20 different values for C ranging from 610 to 57,314.

$$K = \frac{S^3}{C^3} \cdot \frac{1}{(t_d)^4} \qquad \text{Equation 8}$$

Development effort, E, is found by substituting E = 0.4K:

$$E = 0.4\left[\frac{S}{C}\right]^3 \cdot \frac{1}{(t_d)^4} \qquad \text{Equation 9}$$

It can be seen from Equation 9 that the effort E increases as the third power of the size S if the schedule remains constant. For a fixed program size, the effort E increases with the inverse of the fourth power of t_d. This relationship has been disputed by other researchers [Conte et. al. 1986, Kitchenham 1990]. The resulting optimum development schedule is:

$$t_d = 2.4E^{1/3} \qquad \text{Equation 10}$$

Equation 10 agrees substantially with most statistical models used in practice today.

2.2.3 Statistical Models

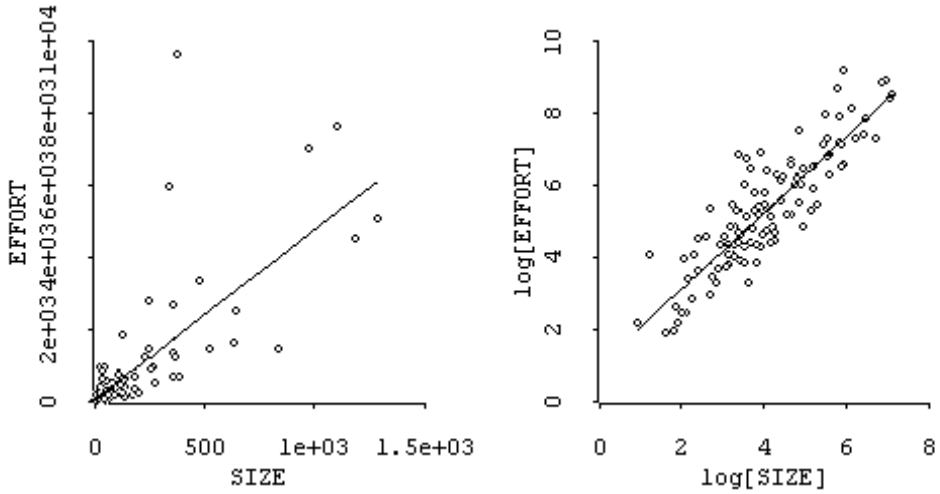
Statistical models use data to derive the values for model coefficients. Regression analysis is used to establish the relationship between model parameters and software development effort. There are two forms of statistical models: linear and non-linear.

Linear statistical models have the form:

$$\text{Effort} = b_0 + \sum_{i=1}^{n} b_i x_i \qquad \text{Equation 11}$$

where x_i are software development factors that are believed to influence effort and b_i are coefficients. There two reasons that models of this form do not work well for estimating software development effort:

1. Empirical evidence shows that the relationship between software development effort and size of the software product is not linear. Figure 4 shows two plots of product size, Adjusted Thousands of Delivered Source Instructions, to development effort, Actual Man Months taken from the database in [Boehm 1981]. Figure 4-A is a plot in linear space. The linear relationship is expressed as: $E = 68.27 + 9.24(\text{Size})$. Figure 4-B is a plot in $\log_e$ space. The nonlinear relationship is expressed as: $E = 1.36(\text{Size})^{1.11}$. The more suitable relationship is obvious.



(A) (B)

Figure 4. Linear vs. Non-Linear

2. As the software product gets bigger effort exhibits a *diseconomy* of scale. This diseconomy of scale with an exponent of 1.11 is shown in Figure 4-B. Economies / diseconomies of scale will be discussed shortly.

Given the evidence of diseconomies of scale linear models are not accurate for modeling effort expenditure. This includes the linear model based on a counting metric called Function Points. The original Function Points was published by Albrecht in 1979 [Albrecht and Gaffney 1983]. This metric consists of counting the number of inputs, outputs, inquiries, interfaces, and logical files from the user's perspective and weighting the counts as simple, average, or complex. The total unadjusted function point count was ad-

justed with 14 complexity characteristics to derive an adjusted function point count, FP. From data presented in [Albrecht and Gaffney 1983] the estimation for effort in person-hours was $E = 54 \cdot FP - 13390$. However, when the data points from the article were plotted the effort in person-hours was found to be of the relation $E = 1.1(FP)^{1.49}$, see Figure 5. This also supports the conclusion that the relationship between size and effort is non-linear.

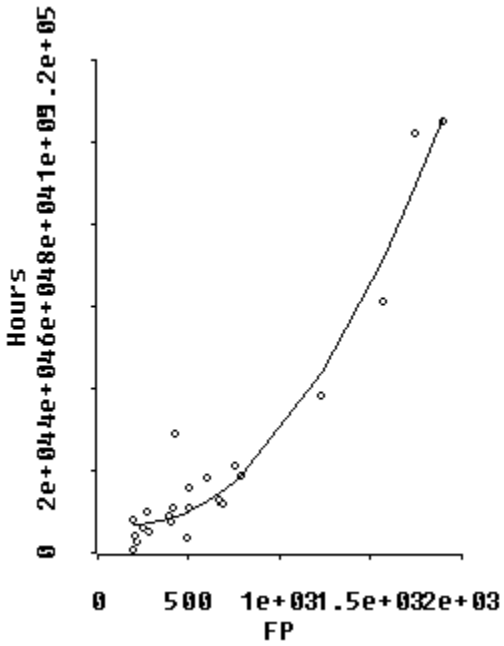


Figure 5. Non-Linear FP Relationship

Non-linear estimation models have the form:

$$Effort = A \cdot (Size)^{b} \qquad \text{Equation 12}$$

where S is size A is a combination of project factors that affect effort. The exponent, b, in non-linear models supports the concept of economies and diseconomies of scale in software development [Banker et al. 1994, Boehm 1981]. Table 2 shows the exponent, b, derived from regression of different data sets for non-linear models [Banker and Kemerer 1989, Boehm 1981, pp.86].

The reasons for economy of scale in software development are [Banker et al. 1994, Boehm 1981]:

- Specialization of labor
- Learning curves
- Software engineering tools
- Diagnostic aids
- Documentation aids

Table 2: Summary of Log-linear Models

Data Set	b	Data Set	b
Yourdon	0.72	COCOMO	1.11
Kemerer	0.85	Frederic	1.18
Walston-Felix	0.91	Phister	1.275
Behrens	0.94	Jones	1.40
Bailey	0.95	Freburger-Basili	1.48
Nelson	0.98	Albrecht	1.49
Herd	1.06	Halstead	1.50
Belady	1.06	Schneider	1.83
Wingfield	1.06		

- Program library aids
- Pre- and post-processors
- Fixed project overhead

The reasons for diseconomies of scale in software development are [Banker et al. 1984, Boehm 1981]:

- More effort to manage the project.
- More extensive testing required to cover increased number of interfaces.
- More time is spent communicating among a larger development team.
- More effort required to design complex of interacting subsystems and then validate that design to requirements.

These models work under the premise that there exists a strong relationship between development effort and software product size. Estimating effort relies on the project being estimated behaving as an *average* of the previous projects in the database. The purpose of using project factors, A, in an estimate is to explain the deviations displayed by the project being estimated from the statistically derived nominal project.

2.2.4 Assessment of Models

Considering the research on Process Maturity effects discussed in Chapter 1 and further explained in Chapter 3, Analogy models are not suitable for this type of research. The model does not give insight into the potential effects of software development process changes. This would make assessing Process Maturity's effect on effort a unqualified estimate.

Models based on theory are not usable for this research because of the aggregation of input parameters and the reliance of an underlying theory to explain and predict effort. Researchers have disagreed with some of the assumptions in these models [Conte et al. 1986, Kitchenham 1990]. It is not clear how these models account for an Iterative software process model where the effort from one build is overlapped with the effort on the next build.

Statistical models are easy to understand. The effect of the model inputs on effort is made understandable by observing the position of the inputs in the mathematical model.

The model is suitable to support this research if a technique can be found to calibrate the inputs and identify / resolve the correlations between inputs.

Statistical models have the disadvantage of possibly producing results that are only valid for the local environment. Another disadvantage to a statistical approach is that as the number of model inputs increases, the amount of data needed to calibrate the model increases (this has to do with the model degrees of freedom).

2.3 Multiple Regression Analysis

Multiple regression analysis is a statistical technique that can be used to analyze the relationship between a single response variable and multiple predictor variables [Hair et al. 1995, Weisberg 1985]. For this research, the response variable is effort, Person Months, and the predictors are factors that influence the effort required to develop a software product. The objectives of multiple regression analysis are to specify the predictor variables in a mathematical equation that will estimate the response variable. Each predictor variable is weighted. The weights denote a variable's relative contribution to the overall prediction of the response.

$$\hat{Y} = B_0 + B_1X_1 + B_2X_2 + \ldots + B_kX_k \qquad \text{Equation 13}$$

where $\hat{Y}$ is the estimated response variable, X_i's are the predictor variables, B_i's are coefficients that act as weights, and k is the number of predictor variables.

As was discussed earlier, linear statistical models are inadequate to model effort expenditure. A non-linear model is needed. A multiplicative model is proposed which can model diseconomies of scale and which can be transformed into a linear model for use in regression analysis.

$$\hat{Y} = A \cdot X_1^{B_1} \cdot X_2^{B_2} \cdot \ldots \cdot X_k^{B_k} \qquad \text{Equation 14}$$

The above model has desirable characteristics that will support this research. It is clear how it works, i.e. the effect of the different input parameters on the final result can be assessed. The simplicity of a model helps reveal the model assumptions and insights about the software process. There is a straight-forward mathematical technique to derive the model exponents (discussed next). The accuracy and fit of the model can be measured. The model can be automated which will simplify the estimation and analysis process.

2.3.1 Log-Log Model

The Log-Log production function, from the field of Econometrics, is a non-linear model that can be transformed into a linear model thus permitting the use of linear regression techniques [Griffiths et al. 1993, pp. 258,277]. The non-linear form is given in Equation 14.

The non-linear model in Equation 14 can be transformed into a linear model by taking the logarithms of both sides of the equation.

$$\ln(\hat{Y}) = B_0 + B_1\ln(X_1) + B_2\ln(X_2) + \ldots + B_k\ln(X_k) \qquad \text{Equation 15}$$

The Bi's were the exponents for the X's in Equation 14 and here they are the coefficients for the X's. In Econometrics they are called *elasticities*. An elasticity represents the percentage change in Y brought about by a percentage change in X, [Griffiths et al. 1993, p. 174]. For instance if B_2 had a value of 1.4 and X_2 changed 10% then Y would change 14%. In this case, it is the percentage change in the Y brought about by a percentage change in the X_i.

$$B_i = \frac{\text{Percentage change in Y}}{\text{Percentage change in Xi}} \qquad \text{Equation 16}$$

which transforms to:

$$\text{Percentage change in Y} = \text{Percentage change in Xi} \cdot B_i \qquad \text{Equation 17}$$

In the above models, B_i, represents the elasticity for the entire population. Since the data for the entire population does not exist, B_i is estimated with b_i. Because b_i is an estimate there is error associated with it and based on this error, there is a prediction interval about b_i in which B_i should reside.

Equation 18 shows a system of transformed equations for n observations and *k* variables. This system is used to derive the values for b_i by minimizing the sum of squared errors. When preforming the regression analysis, the $\hat{Y}$ are substituted with the actual effort observed, Y, on a project. The X_i's are the actual observed values for the predictor variables.

$$\begin{aligned}
\ln(\hat{Y}_1) &= \ln(b_0) + b_1\ln(X_{1,1}) + b_2\ln(X_{1,2}) + \ldots + b_k\ln(X_{1,k}) \\
\ln(\hat{Y}_2) &= \ln(b_0) + b_1\ln(X_{2,1}) + b_2\ln(X_{2,2}) + \ldots + b_k\ln(X_{2,k}) \\
\ln(\hat{Y}_3) &= \ln(b_0) + b_1\ln(X_{3,1}) + b_2\ln(X_{3,2}) + \ldots + b_k\ln(X_{3,k}) \\
&\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \\
\ln(\hat{Y}_n) &= \ln(b_0) + b_1\ln(X_{n,1}) + b_2\ln(X_{n,2}) + \ldots + b_k\ln(X_{n,k})
\end{aligned} \qquad \text{Equation 18}$$

The minimum number of observations, n, required for regression has to be at least k+1.

Every model of real-world phenomena has error. The estimated standard error (est s.e.), or standard deviation (SD), for this model can be found by comparing the predicted value, $\hat{Y}$, to the actual value, Y, for each of *i* observations where $1 \le i \le n$ [Griffiths et al. 1993, pg.23]:

$$\text{est s.e.} = \sqrt{\frac{\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}{n - k - 1}} \qquad \text{Equation 19}$$

With the standard error, a prediction interval can be found. The *t* distribution is used in constructing the interval because the population standard error of estimate is unknown. However the response variable is assumed to be normally distributed in the population. The value for *t* is found in a table for Student's *t* distribution and depends on degrees of freedom and α. The degrees of freedom is the number of observations minus the number of independent variables minus one: *n - k - 1*. For this research, α is set to 0.025 which gives a 95% prediction interval.

$$\text{Prediction Interval (95\%)} = (\hat{Y}) \pm (t_{(df,\alpha)} \cdot \text{est s.e.}) \qquad \text{Equation 20}$$

2.3.2 Hypothesis Testing

Hypothesis testing for this research amounts to determining if a coefficient is non-zero. A non-zero coefficient would show that the related predictor variable (e.g. process maturity level) does affect effort. The null hypothesis, H_0, and the alternative hypothesis, H_1, are stated as follows:

$$H_0: B_i = 0$$
$$H_1: B_i \neq 0 \qquad \text{Equation 21}$$

The objective of the analysis is to reject the null hypothesis at the given confidence level, thereby showing that the predictor does effect effort.

In reality though, B_i, is not known. It can only be estimated using b_i. To successfully conclude that b_i's predictor variable does affect effort, b_i, must not be equal to zero. A test must be performed to check if zero is within the estimation interval. This is called a t-test which checks for Type I errors. A Type I error is the probability of incorrectly rejecting the null hypothesis when a correlation between the predictor variable and the response does not really exist.

For the t-test a t-value is computed from the estimated coefficient, b_i, and the coefficient's standard error, $s.e._i$. The standard error for the coefficient is the square root of the coefficient's variance. Equation 22 shows the t-value computation [Weisberg 1985, pg.20]. The difference between the estimated and actual coefficient is normalized by the standard error of the coefficient. Since B_i is not known, the value from the null hypothesis is used instead, zero. It can be seen that the t-test represents a signal to noise ratio. The stronger the signal coming from a predictor variable the smaller the estimated standard error. But if there is a lot of noise or standard error in the signal then the influence of the predictor might remain unknown.

$$\text{t-value}_i = \frac{b_i - B_i}{\text{est s.e.}(b_i)} \qquad \text{Equation 22}$$

The t-value is compared to Student's t distribution to determine if it is significant, Equation 23. The t distribution is used because the population standard error of the estimate for B_i is unknown and the response variable is assumed to be normally distributed in the population. The value for t is found in a table for Student's t distribution and depends on degrees of freedom and the confidence level α. The degrees of freedom (df) is the number of observations, n, minus the number of independent predictor variables, k, plus the intercept term: df = n - (k + 1). The symbol α represents the probability of committing a Type 1 error in Hypothesis testing [Griffiths et al. 1993, pg.136]. With a 95% level of confidence and as degrees of freedom get very large, the t distribution value is 1.96 [Griffiths et al. 1993, p. 845]. If the absolute value of the computed t-value exceeds the t distribution value then the coefficient is considered significant with a 95% confidence level. For a 90% level of confidence, the t distribution value is 1.65. This lower value is easier to achieve but there is a little more uncertainty as to whether a Type I error has occurred.

$$|\text{t-value}| \geq t_{(df,\,\alpha)} \qquad \text{Equation 23}$$

2.3.3 Regression Model Assumptions and Collinearity

Regression models must satisfy five assumptions to be valid in their results:

1. The independent variables and the dependent variables have a linear relationship. The linear relationship of the Log-Log model is expressed by Equation 15.
2. The dependent variable is a continuous random variable and the independent variable are set at various values and are not random.
3. The variances of the dependent variable are equally distributed given various combinations of the independent variables.
4. Successive observed values of the dependent variable are uncorrelated.
5. The distribution of the sampling error, e_i, in the regression model is normal.

The regression model is used in this research to depict the effects of the independent variables on the dependent variable. If the independent variables are not linearly independent from each other, determining the contribution of each independent variable will be difficult because the effects of the independent variables are mixed. Thus the regression coefficients may be incorrectly estimated. Interdependence of independent variables is called *multicollinearity*.

Figure 6 shows a Venn diagram which serves as a conceptual model for regression. The box represents the dependent variable, the effort required to develop a software product. Each circle is an independent variable used to estimate effort. The size of the circle is proportional to the amount of correlation between the independent variable and the dependent variable. No intersection of the circles means the independent variables are not correlated among themselves. An intersection is an example of collinearity between the variables, see (X_6, X_9) and (X_3, X_{10}, X_7) in the figure. The amount of intersection can be

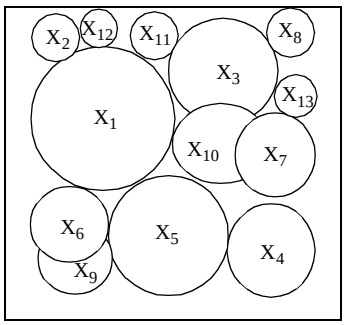


Figure 6. Multicollinearity

thought of as the degree of collinearity between the variables which ranges from -1 to 1. A collinearity of zero (0) means no overlap in independent variable effect on the dependent variable. A value of -1 means the independent variables are inversely correlated and a value of 1 means they are directly correlated. The space between the circles is the amount of variance unaccounted for by the independent variables. Collinearity may be due to the combined effect of two or more other independent variables.

Possible solutions to handle the collinearity are:

- Omit or combine the highly collinear independent variables and find other variables to use in the model that are not collinear.
- If the variables are truly thought to be independent, collect more data (the collinearity results may come from noise in the data).

2.3.4 Model Evaluation

A regression model predicts a response variable's value, $\hat{Y}$, based on the assumption that the value is the same as the average value from a set of observations in the database. Deviations from the average or mean observed value, $\overline{Y}$, are explained by the predictor variables. These variables are used to adjust the average to be close to the actual observed value.

Figure 7 shows a data point and a regression line (solid) for a single predictor and response regression model. The average predictor variable value, $\overline{X}$, and the average response variable value, $\overline{Y}$, define a point the regression line must pass through. This point represents the average observation. A point, (X_i, Y_i) in the figure sits above the regression line. The distance from the point to the regression line is unexplained variation and is an error. The sum of the errors from all observations is squared and called the Sum of Squared Error (SSE). The distance from the regression line to the average line for the response variable, $\overline{Y}$, is explained variance and it is due to the regression line. The sum of the explained variations from all observations are squared and called Sum of Squared Regression (SSR).

The distance from the point (X_i, Y_i) to the average line for the response variable is called the Total Sum of Square, $SST = SSR + SSE$.

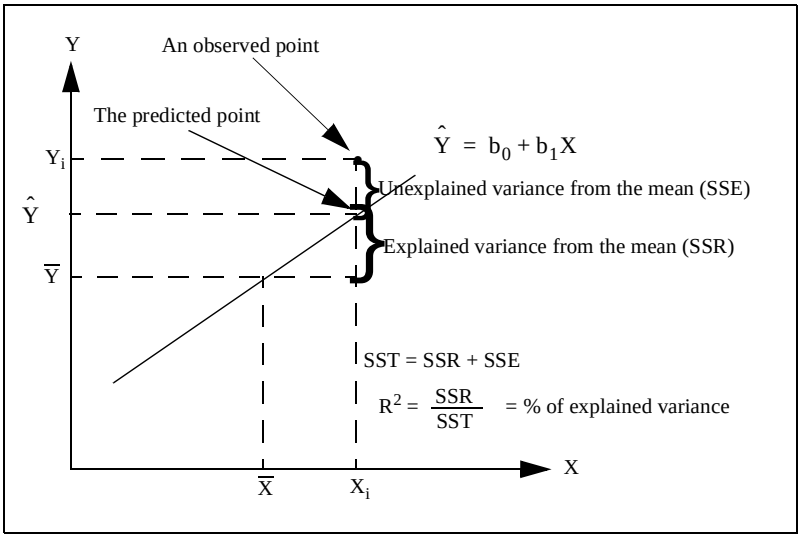


Figure 7. Explained and Unexplained Variance from the Mean

There are four criterion used to judge how well a model fits the data and how well it will forecast. Adj-R^2 and the standard error statistics give an indication of the "goodness" of fit of the model to the data. P.E. and Pred(X) give an indication of the model's accuracy.

R^2 is called the coefficient of determination and it ranges between 0.0 and 1.0. R^2 denotes the percentage of variance explained by the predictor variables used in the regression analysis, Equation 24. An R^2 near 1.0 indicates that almost all of the variability in the predicted response variable is explained by the model and that inclusion of additional predictors variables in the model is not likely to improve the model.

$$R^2 = \frac{\sum_{i=1}^{n} (\hat{Y} - \overline{Y})_i^2}{\sum_{i=1}^{n} (Y - \overline{Y})_i^2} \qquad \text{Equation 24}$$

It is also known that the more predictors added to a model, the higher the R^2. Adjusted R^2, Adj-R^2, is a more realistic indicator of the model "goodness of fit" because it is adjusted for the number of parameters in the model. Adjusted R^2 is always less than R^2 [Weisberg 1985, pg.217]. If the response values in the dataset vary greatly about the mean

then the model estimated values of the response can vary greatly about the mean of PM and still produce a high R^2.

$$\text{Adj-}R^2 = 1 - \left(\frac{n-1}{n-k-1}\right)(1 - R^2) \qquad \text{Equation 25}$$

The standard error for the model expresses the "noise" that is in the data. Standard error is given in Equation 19. It can also be expressed in terms of SSE.

$$\text{est s.e.} = \sqrt{\frac{SEE}{n-k-1}} \qquad \text{Equation 26}$$

Proportional Error (PE) is a measure of relative error. As the estimates become larger for larger projects, the residual (or pure error) is normalized for project size. Equation 27 shows this normalization.

$$\text{P.E.} = \begin{cases} [\hat{Y} \div Y] - 1 \text{ when } \hat{Y} - Y \geq 0 \\ -[Y \div \hat{Y}] + 1 \text{ when } \hat{Y} - Y < 0 \end{cases} \qquad \text{Equation 27}$$

There have been previous evaluations of cost models that used Relative Error, $RE = (\hat{Y} - Y)/Y$, as a measure of prediction accuracy [Boehm 1981, Conte et. al. 1986]. Usage of this error statistic shows that it is bounded at -1.0 in the negative numbers and it can extend to infinity in the positive numbers. This presents misleading results and visually it presents a biased graph. Magnitude Relative Error has also been proposed for use [Conte et. al. 1986 and Kemerer 1989?] but it is more misleading because it folds the negative errors between -1.0 and 0 into the interval between 0 and 1.0. PE is used because it is symmetric about 0 and extends to infinity in both directions.

The last evaluation criterion is the percentage of predictions that fall within X% of the actuals, denoted as Pred(X) [Conte et al. 1986, pg.173]. The models are evaluated at Pred(20) and Pred(40) which is done be counting the number of PE's less than or equal to 0.20 and 0.40 respectively and dividing by the number of projects.

Chapter 3

SW-CMM CASE STUDIES AND AVAILABLE EFFORT ESTIMATION MODELS

This chapter is a survey of "what is out there." It has two sections. The first section examines the literature on the benefits of using the Software Capability Maturity Model to increase process maturity (this model was discussed in section 2.1 on page 3). Both case studies and studies of case studies are reviewed. The second section surveys cost models available either commercially or from the literature to determine if they account for the effects of Process Maturity as inputs for estimation of software development effort.

3.1 The Capability Maturity Model for Software

There have been many reports on the benefits of adopting the Software Engineering Institute (SEI) Software Capability Maturity Model (SW-CMM). This section surveys the most authoritative articles for the cost and benefits of increasing Process Maturity.

3.1.1 Institute for Defense Analysis

The Institute for Defense Analysis (IDA) performed a study [Springsteen et al. 1992] for the U.S. Department of Defense that presented quantitative and qualitative data on the SW-CMM and that compared the SW-CMM to other process maturity models (these comparisons will be discussed later).

IDA's review of cost estimation models found disagreement among their proprietors with respect to the effects of the SW-CMM. Capers Jones based his assessment on his Checkpoint Model from Software Productivity Research with its associated database of 3000 projects. His prediction was that quality of software would peak at SW-CMM Level 3 and the productivity would peak at Level 3 and decline for Levels 4 and 5. His analysis was an extrapolation based on the very small sample of projects at the time with high SW-CMM levels. For example the IBM-Houston Space Shuttle software project was assessed at Level 5, but its productivity was not high, due in large measure to its safety-critical nature.

Another expert, Larry Putnam, based his assessment of SW-CMM effects on his Productivity Analysis Database System with its database of 1500 projects. He assumed that there was a similarity between the Productivity Index (based on development size, effort, and time) and SW-CMM Levels. Higher levels of Process Maturity resulted in higher levels of productivity. However, the Productivity Index does not account for specific software practices such as those specified in the SW-CMM and it may not be correlated with the SW-

CMM Levels. For example, many of the projects his database indicated were Level 5 may have had a high Productivity Index because of the low complexity of their applications.

IDA surveyed 480 users of the SW-CMM. 88% of the users that had performed internal process maturity assessments felt that it was useful for identifying areas that needed improvement. 68% of the users that had an external evaluation thought that it was a viable contract selection criterion. 55 companies were surveyed and the majority of them had no definitive measurement results although several companies had significant process improvement efforts.

The IDA assessment of the available data on increasing Process Maturity was favorable:

- Limited case studies indicated positive return on investment and improved quality
- There was not enough information to separate out other factors that might have been improved
- Many firms were just starting data collection on the effects of process improvement

3.1.2 Hughes Ground Systems Group

The Software Engineering Division at Hughes Aircraft in Fullerton, Ca, spent $400,000 and 2 years improving their Process Maturity from Level 2 to Level 3 [Humphrey et al. 1991]. Several observations are made in this paper. It takes management commitment to survive the investment, risk, time, and pain of cultural change that occurred during the transition period. Achievements feed on themselves and when the whole organization buys into the improvement process, it gains a sense of esprit de corps. Increasing maturity levels reduced risk in meeting planned costs and schedules. The reduction in planned versus actual budgets saved Hughes about $2 million annually—a short term gain. There were fewer overtime hours, fewer gut-wrenching problems, and a more stable work environment.

This report shows that in addition to the activities required by the SW-CMM to move from maturity Level 2 to Level 3 there were collateral benefits that helped improve their software capability. “Esprit de corps” affects the effort people put into a product—they are more motivated. The work environment became more stable. Technology insertion became a visible, controlled activity. These effects need to be accounted for in analyzing the difference made by implementing KPAs from the SW-CMM, i.e. some of the gains may have been achieved via other people and technology improvements without using the SW-CMM.

3.1.3 Raytheon

The Raytheon Software Systems Laboratory in the Equipment Division had the goal of transitioning from SW-CMM Maturity Level 1 to Level 3 [Dion 1993]. This initiative took approximately 5 years and the Division invested almost $1 million. A sequence of three steps was cyclically followed to manage change:

1. Process-stabilization where elements of the process were identified and institutionalized progressively across all projects.

2. Process-control where projects are measured to gather significant data which is analyzed on how to control the process.
3. Process-change where processes are adjusted and technology is transitioned into the process.

The initiative had top management support in that the manager of the Software Systems Laboratory was the chair of the steering committee that provided direction and oversight. Four groups were formed to assist in implementing the infrastructure to support maturity level transition: policy and procedures group, training group, tools and methods group, and process database group.

Raytheon measured the effects of increasing their Process Maturity by looking at the cost of performance (the cost of doing it right the first time) and the cost of quality: appraisal (the cost of testing for faults), rework (cost of fixing defects), and prevention (the cost of preventing the fault from getting into the product). This approach is based on [Crosby 1984]. The most notable benefit of moving to a higher maturity level is the savings *due to the reduction in rework*. It is estimated that $15.8 million was saved from August 1988 through November 1992. Rework savings were achieved at the expense of an increase in the early life-cycle activities (design and coding) to find the errors early before they were discovered in integration and required fixes and retesting. The main practice that was changed was that informal inspections were replaced by formal inspections.

In additional to reduction in rework, other collateral benefits were realized. Work conditions improved (less nights and weekends), job satisfaction increased, less schedule erosion, and higher levels of interpersonal communication. The areas of work environment and team cohesion helped increase the productivity in software development production. Again, however, there was no way to separate out SW-CMM-related effects from the effects of concurrent people and technology improvements.

3.1.4 Schlumberger

Schlumberger's Laboratory for Computer Science has the charter in part to help software engineers improve software productivity and product quality. Initially, an evaluation was performed using assessment techniques from the SEI. The evaluation revealed that improvement was needed in project management, process definition and control, and project planning and control. They reported that three components drive improvement in software development productivity: process, people, and technology. For people, training them in project management and peer reviews has a very beneficial effect. For tools, evaluating and disseminating results of the evaluation on CASE tools, C++ environments, and Requirement Management tools make tool adoption more efficient. Collaboration was seen as the most important process improvement. Training was very important as well.

The positive results reported were influenced by the three components: process, people, and technology. The report did not separate out the individual effects of each.

3.1.5 Oklahoma City Air Logistics Center

The Center hired a consultant, Software Productivity Research, to determine the economic benefit of Software Process Improvement [Butler 1995]. Four projects that pro-

duced program sets for an engine control and avionics for three airplanes were studied. The first project, the baseline project, was started in March 1986 and ended in May 1988. The last project studied started in June 1992 and ended in March 1995. The Center was rated a Level 1 in 1990 and a Level 2 in 1992.

An important effect mentioned in the report was the formation of a Management Steering Team and a Software Process Engineering Group. These groups met once a month to clear any problems that might impede the progress of process improvement. They address both SW-CMM and non-SW-CMM issues.

The results reported are a 7.5 to 1 Return on Investment (ROI) and a factor of ten improvement in productivity. The ROI figure was derived by comparing the baseline project to the three subsequent projects. The additional amount the three projects would have cost had there not been any improvements in productivity were used to compare to the baseline project: 7.5 to 1.

The factor of ten productivity improvement gain is attributed to both process and technology improvement and the effect of each could not be separated out. It is difficult to achieve more than a factor of 2 improvement through pure process improvements; higher factors generally involve software reuse or very high level languages [Boehm 1993]. This does not help in understanding how process improvement alone affects productivity.

3.1.6 Software Engineering Institute

A report was published from the Software Engineering Institute [Herbsleb et al. 1997] that gave some results of the effects of software process improvement on organizations. They looked at published case studies (some of which are reported here) and surveyed organizations that had appraisals within one to three years from the date of the report. One hundred sixty seven questionnaires were sent out and one hundred thirty eight returned responses. To assess the effect of Process Maturity the survey sought information on how long it took to change SW-CMM Levels, how much did it cost, and how did it benefit business?

The range for moving from SW-CMM Level 1 to Level 2 was 1.5 to 2 years. From Level 2 to Level 3 the time to move ranged from 17 to 31 months. The range for cost per software engineer was $490 to $2004.

There were five benefits that had four possible responses: excellent, good, fair, or poor. The benefits being surveyed were product quality, customer satisfaction, ability to meet schedule, ability to meet budget, and staff morale. All of the benefits increased with maturity level.

Table 1 in the report shows a productivity gain per SW-CMM Level of 25%. This data is based on four observations. It is noted in the report that a detailed study by Krishnan on the relationship between Process Maturity and software quality of a Fortune 100 Company showed a significant increase in quality but no direct evidence of an effect on cost [Krishnan 1996].

3.1.7 LOGOS International Inc.

A study was performed by LOGOS International for the Air Force on the Return on Investment (ROI) for increasing maturity level using the SW-CMM [Brodman and Johnson 1995]. Questionnaires and interviews were used to survey 33 companies as well as a literature search was conducted. The investigator found differing definitions for ROI. The textbook definition is the amount returned in realized gains to the amount invested to improve. The Government respondents in the survey looked at investment in terms of cost. Their definition was the cost of savings due to reduced operating expenses to the cost of investing in new technology. The Industry respondents focused on effort. Their definition was effort saved to effort invested in improvement. Companies want to stay within budget, meet quality goals, meet requirements, and build a maintainable product.

Respondents noted non-SW-CMM changes. These were in attitude, less overtime, less turnover, and an improved competitive edge. Three data points on change in effort required to develop a product showed effort decreased. There were many reported increases in productivity: 10-20%, 90-100%, 50%, 15-20%, 5%, 130%, 12%, 6.3%, and 30%.

This research shows that case studies that report on improvement in ROI due to increased Process Maturity may not be comparable. This is due to the different definitions for ROI. Also the range of increases in productivity, 10-130%, make it difficult to pin down how much Process Maturity affects productivity.

3.1.8 DACS Study

A state of the art report by Kaman Sciences Corp. done for Rome Laboratory reviewed the literature on SW-CMM-based improvements and the benefits of software reuse, inspections, and Cleanroom Software Engineering [McGibbon 1996]. This report broadened the view of Software Process Improvement to include developmental technologies. It also took the unique approach of developing a Software Process Improvement model in a spreadsheet. The model is used to show ROI, benefits of inspections, software reuse and Cleanroom Software Engineering. The model uses COCOMO [Boehm 1981] and a quantification of seven stages an organization moves through when increasing process maturity.

This report is different in that it attempts to use raw numbers in the literature to construct a model. The literature shows a wide range of information some of which is not comparable between case studies. The seven stages used in the model have no correlation to the SW-CMM. The model is not based on collected data.

3.1.9 SEI Capability Maturity Model's Impact on Contractors

This article acknowledges the successes of the SW-CMM discussed above [Saiedian and Kuzara 1995]. But it also points out that the SW-CMM assumes that major software development problems are managerial and not technical. It notes that the SW-CMM does not directly address expertise in an application domain, advocate specific tools, methods, or software technologies, or address issues related to human resources such as how to select, hire, motivate, and retain competent people. It does not address issues related to con-

current engineering, teamwork, change management, or systems engineering. These claims are acknowledged in [Paulk et al. 1993].

The SW-CMM has been impressed on industry by government and defense-oriented software acquisitions. A dilemma that contractors face is that moving from level to level can cost hundreds of thousands of dollars but the government selects bidders using lowest development cost as a significant criterion (a problem the SW-CMM was commissioned to help resolve). How do organizations pay for improving process maturity? How do they justify their choice of investments?

3.1.10 Other Assessment Criteria for Process Maturity

In addition to the SEI Capability Maturity Model which was first published as the Process Maturity Model in 1987 [Humphrey and Sweet 1987] there are other software development models / criteria to assess process maturity. These assessments are from a company called Software Productivity Research, the Air Force's Software Development Capability/Capacity Review and the ISO-9001 and ISO-9000-3 [Springsteen et al. 1992, Paulk 1995b].

3.1.10.1 Software Productivity Research

The Software Productivity Research (SPR) assessment consists of approximately 400 questions that are applied to individual projects within an organization [Springsteen et al. 1992]. The assessment identifies strength and weaknesses at both the project and, using combined project data, at the organization level. The project assessments are compared to other projects within the same organization, with combined-project profiles of the organizations, and with a composite profile of the software industry as a whole. The information collected enables process improvement actions to be taken at the project level and the detail of information that is tracked makes improvement more easily observable. The assessment covers areas about the physical environment provided for software developers, experience level of key staff members, development methodologies used, automated tools employed, testing techniques applied, and the degree of design and code reuse achieved. The areas covered by the SPR assessment have some overlap with the SW-CMM. The models will be compared below.

The SPR assessment has some weaknesses. It is not based on a mathematical model of software development but is based on analogy. The strength of the evaluation is in the comparison to data that was collected from past assessments and this information kept proprietary. Use of analogy does not provide guidance on how to prioritize process improvement activities. The SPR assessment's focus on individual projects does not capture organization-level issues that influence software development such as training, standards/ procedures, or for the parties responsible for process improvement.

3.1.10.2 Software Development Capability/Capacity Review

The Software Development Capability/Capacity Review (SDC/CR) assessment consists of 450 essay questions. Its purpose is to assess the offeror's capability and capacity

to develop software as required for a particular software product [Springsteen et al. 1992]. The assessment looks at eight categories: management approach, management tools, development process, personnel resources, Ada, flight critical software, AI, complex hardware development. The assessment includes site visits where the contractor explains their software development approach. This assessment covers categories not in the SW-CMM.

The weakness of this assessment method is that there is little guidance for process improvement. It is focused strongly on use for source selection. This assessment is not based on a model but on criteria for rating the essay responses. Poor performance in an area does not indicate what process improvements should be made.

3.1.10.3 ISO-9001 and ISO-9000-3

ISO-9001 is the standard in the ISO-9000 series that pertains to software development and maintenance. It identifies the minimal requirements for a quality system. It is used to ensure the supplier conforms to specified requirements during several stages of development, including design, development, production, installation, and servicing. The ISO-9000-3 provides guidelines for applying ISO-9001 to the development, supply, and maintenance of software. Assessments are done by a trained and certified evaluation team.

The ISO-9001 does not address personnel or software development technology capabilities. It is focused on the methods, techniques, and tools that a process would have to use to produce a quality product. There is some overlap with the SW-CMM but the focus of the SW-CMM is continuous process improvement with which higher quality is a by-product [Paulk 1995b].

3.1.10.4 Comparison of Assessment Criteria

Table 3 which was extended from [Springsteen et al. 1992] uses nine attributes to show a summary comparison of the different process maturity assessment criteria discussed above. Project tailoring is for criteria that assess the project's ability to meet the product requirements of type, scope, experience, budget, schedule, and size. Project personnel is for criteria that assess staffing resources, experience, and training. Project management is for criteria that assess project structure, estimation, tracking and commitment. Methods and tools are for criteria that assess tool and method support for requirements, design, support and development tools. Product and technology constraints are for criteria that assess the ability of the project to work within hardware, language, required reuse and customer furnished equipment constraints. Quality and configuration control is for criteria that assess a project's quality assurance, configuration management, and review procedures. Project measurement data is for criteria that assess quantitative measure of progress, quality, and productivity. Organization process support is for criteria that assess standards, training, and planning. Organization technology support is for criteria that assess the infusion of tools and software development technology into the process. In Table 3, the black circle is the highest rating and the hollow circle is the lowest rating.

Table 3: Assessment Criteria Comparison

Attributes	SW-CMM	SPR	SDC/CR	ISO-9001
Project Tailoring	●	●	◖	●
Project Personnel	○	◖	●	○
Project Management	●	○	◖	●
Methods and Tools	○	◖	◖	○
Product and Technology Constraints	○	◖	◖	◖
Project Quality and Configuration Control	●	◖	●	●
Project Measurement Data	●	◖	○	●
Organization Process Support	●	○	○	◖
Organization Technology Support	◖	○	○	○

3.2 Available Effort Estimation Models

This section is a survey of cost estimation models for their treatment of the effects of Process Maturity in the model. This is done by examining the model inputs and reasoning about how Process Maturity influences the model.

3.2.1 Wideband Delphi

This method seeks to gain consensus on an effort estimation by a group of experts [Boehm 1981]. The process works by having a moderator disseminate software requirements and an effort estimation form to a selected group of experts. A meeting is held where the experts discuss estimation issues. Then each expert fills out the estimation form. The moderator collects and summarizes the estimates. Another meeting is called and the anonymous differences in estimating points is discussed. The experts fill out the estimation form again and the moderator collects and summarizes the estimates. The process is repeated until there is convergence on an estimate. Everyone has a partial view of the total effort required for development. The Delphi process shares those views.

This estimation technique does perhaps consider Process Maturity by having an expert use the performance from a previous project that was at a specific level. It may not be considered though if the expert does not consciously make a comparison of the intended maturity level of the new project to the level of the past project being used as the basis of estimate.

3.2.2 Work Breakdown Structure

The Work Breakdown Structure (WBS) method of estimation is based on breaking down the work to be done into smaller and smaller subsystems until the individual tasks are

identified [Boehm 1981]. For each task, either a database is consulted or an expert makes an estimate on the amount of effort required to complete the task. A process of "rolling up" all of the task estimates into their respective subsystems then up to the system level produces an overall estimate.

If there are defined processes, the tasks specified in those processes should be in the tasking WBS. The effect of process maturity cannot be seen when using the WBS but its effects can be accounted.

3.2.3 Checkpoint

Checkpoint is a knowledge based software estimation and assessment tool. It contains its own knowledge base of thousands of software projects from different application domains. The projects in the knowledge base represent new and maintenance projects. It uses Function Points or Feature Points to measure the size of a software project.

Checkpoint considers Project Management factors in assessing a project. The inputs focus on management experience, management methods and tools, managerial and technical cohesiveness, and measurement activity. There are process inputs as well that include development methods, quality assurance, and testing [SPR1994]. These inputs capture some of the focus of the SW-CMM but do not consider Key Process Areas such as Peer Reviews or Intergroup Coordination.

3.2.4 SLIM

SLIM (Software Lifecycle Model) is a software cost, schedule, risk, and reliability estimation tool for project planning, project control, and risk analysis. It is based on the theoretical model discussed in section 2.2.2 on page 7. SLIM uses a "Productivity Index" to encompass many factors including management influence on the project; development methods; development tools, techniques, and aids; skills and experience of the development team; available of resources; and complexity of the application [SLIM 1995]. There is not a direct input for Process Maturity. However the tool can be calibrated to local conditions which would reflect any influence of Process Maturity on project data.

3.2.5 Jensen Model

Randy Jensen proposed a model that is similar to the theoretical model discussed in section 2.2.2 on page 7 [Jensen 1984]. He proposed the following:

$$S = C_{te} T K^{1/2} \qquad \text{Equation 28}$$

Solving for development effort, E, gives:

$$E = 0.4 \frac{S^2}{C_{te}} \cdot \frac{1}{T^2} \qquad \text{Equation 29}$$

A different form of the technology constant is used, called the effective technology constant, C_{te}. This constant consists of a basic technology constant and the product of a

number environmental adjustment factors (this follows the form of the COCOMO Cost Model to be discussed later). The Environmental adjustment factors take into account product, personnel and computer factors that affect effort. Management practices are not considered.

3.2.6 SEER-SEM

SEER-SEM (System Evaluation and Estimation of Resources - Software Estimation Model) is a software cost, schedule and risk estimation tool that address all DOD software standards and requirements. Software issues such as Ada, DOD Standard 2167A, security and others are specifically supported by the model. A knowledge base developed from thousands of DOD projects are an integral part of the model.

This model does have an input parameter called Process Improvement [SEER-SEM 1994]. It captures the amount of effort spent on improvement activities on the next project. Improvement is defined as implementing modern programming practices such as Object Oriented design or Concurrent Engineering. Changing ratings for the input parameter also means changing SEI levels.

This cost model has Process Maturity as an input. This is the only commercial model reviewed with this input. However, because it is a commercial product its formulas are proprietary.

3.2.7 Softcost

The Softcost model attempts to incorporate the good points of a number of different models [Tausworthe 1981]. The model has 68 parameters related to productivity, duration, staffing level, documentation and computer resources. The outputs are effort, schedule broken down into a standard Work Breakdown Structure, staffing level, pages of documentation, and CPU requirements. The model uses management reviews as model inputs. Because the model was created for use at the Jet Propulsion Laboratory, many process factors aggregated with other inputs.

The Softcost model was adopted and extended to a new model, Softcost-R [Reifer et al. 1989]. This model takes process factors as inputs. Those factors include degree of standardization, lifecycle coverage, scope of support, use of modern software methods, use of peer reviews, use of software tools/environment, tool/environment stability, and geographical co-location. This model comes closer to capturing the effects of Process Maturity via the SW-CMM on development effort. It considers peer reviews as a direct input, one of the Software CMM's Key Process Areas. There are other KPAs that are not considered, e.g. project planning and tracking, quantitative process management.

3.2.8 Estimacs

This model uses a size measure similar to function points. There are 25 input parameters in the following groups: size, product, environment, personnel, project, and user. Outputs are effort, schedule, staffing level, and risk assessment. At this time the model was

reviewed, none of the inputs had a direct correlation to SW-CMM Maturity levels. The model is proprietary and the internal details are not available [Rubin 1983].

3.2.9 PRICE S

PRICE S (Parametric Review of Information for Costing and Evaluation Software) estimates size, costs, and schedules for design, programming, integration, testing, and support. The key inputs to PRICE S are software function to establish the size of the program; productivity factor that includes such items as skill levels, experience, productivity, and efficiency as related to an activity such as development; complexity which defines the degree of difficulty; platform which characterizes the operational and reliability requirements; application to capture coding difficulty; and design / code inventory that defines the amount of new design and new code required. The productivity index must be determined from local projects before the model is used. The model can only be used in an environment for which it was calibrated. Process Maturity effects are aggregated in the Productivity Index and their individual influence is not identifiable [PRICE S 1993].

3.2.10 Meta-Model

The Meta-Model is a non-linear model of the form in Equation 30 [Bailey and Basili 1981]. Using data from NASA Goddard Space Flight Center, the following coefficients were obtained:

$$\hat{E} = 3.5 + 0.73S^{1.16} \qquad \text{Equation 30}$$

Equation 30 is called the background equation and predicts effort with the assumption the project under examination behaves as an average of the previous projects in the database. The difference between this project and the historical ones is explained by project factors. The relationship of the project factors, background equation and predicted effort, E, is:

$$\text{Effort} = \begin{cases} (1 + F)\hat{E} \\ \hat{E} / |1 + F| \end{cases} \qquad \text{Equation 31}$$

F is the multiplicative adjustment factor which is derived by regression on the residuals from Equation 30:

$$F = b_0 + b_1 METH + b_2 CPLX + b_3 EXP \qquad \text{Equation 32}$$

METH was an assessment of the methodology, CPLX was an assessment of the cumulative complexity, and EXP was an assessment of the cumulative experience. Each of the characteristics in the categories of methodology, complexity, and experience are rated on a scale of 0 to 5 and summed. The adjustment factor, F, and the background equation, E, are combined in Equation 31 to obtain an estimate of effort.

This model works well on its calibration data set. It was intended not as a general prediction model but one to be adapted to local development conditions. The model cur-

Table 4. Meta-Model Factors

Methodology (METH)	Complexity (CPLX)	Experience (EXP)
Tree charts	Customer interface cplx.	Programmer qualifications
Top down design	Customer-init. design changes	Pgmr. machine exp.
Design formalisms	Application process cplx	Pgmr. language exp.
Formal documentation	Program flow cplx	Pgmr application exp.
Code reading	Internal communication cplx	Team cohesion
Chief programmer teams	External communication cplx	
Formal test plans	Database cplx	
Unit development folders		
Formal training		

rently does not consider Process Maturity as defined by the SW-CMM but it could be modified to accept it as input. The only drawback in using this model is that it uses non-linear regression to derive the background equation coefficients. This is a mathematically difficult technique requiring iteration and initial estimates of coefficients.

3.2.11 COCOMO

COCOMO (Constructive Cost Model) [Boehm 1981] is a set of three models: basic, intermediate, and detailed. The Intermediate COCOMO model estimates Person Months (PM) of effort. It takes as input the estimated size of the software product in thousands of Delivered Source Instructions (KDSI) adjusted for code reuse, the project development mode, *b*, and 15 cost drivers. The development mode can take only three values, {1.05, 1.12, 1.20}, which reflect the difficulty of the development. The estimate is adjusted by factors, called *cost drivers*, that influence the effort to produce the software product, Table 5. Cost drivers have up to six levels of rating: Very Low, Low, Nominal, High, Very High, and Extra High. The estimated effort in Person Months is given as:

$$PM_{estimated} = A(Size)^{b} \prod_{i=1}^{15} (Cost\ Driver)_i \qquad \text{Equation 33}$$

The model does not have a cost driver called Process Maturity but there is a cost driver called “Use of modern programming practices” (MODP). This cost driver is characteristic of organizations that have higher maturity levels. The COCOMO II model is an updated version of COCOMO and it does have Process Maturity as a model input. It is discussed next.

Table 5: COCOMO Cost Drivers

Category	Cost Driver	Symbol	i
Product	Required software reliability	RELY	1
	Database size	DATA	2
	Product complexity	CPLX	3
Platform	Execution time constraint	TIME	4
	Main storage constraint	STOR	5
	Virtual machine volatility	VIRT	6
	Computer turnaround time	TURN	7
Personnel	Analyst capability	ACAP	8
	Applications experience	AEXP	9
	Programmer capability	PCAP	10
	Virtual machine experience	VEXP	11
	Programming language experience	LEXP	12
Project	Use of modern programming practices	MODP	13
	Use of software tools	TOOL	14
	Required development schedule	SCED	15

3.2.12 COCOMO II

The COCOMO II model is an update to the previous COCOMO models [Boehm et al. 1995]. There are three models that comprise the COCOMO II model: Application Composition, Early Design, and Post-Architecture. The Post-Architecture model has the form:

$$PM = A \cdot (Size)^{\left[1.01 + \sum_{j=1}^{5} SF_j\right]} \cdot \prod_{i=1}^{17} EM_i \qquad \text{Equation 34}$$

This is a non-linear model (see the model explanation in section 2.2.3 on page 9) that has an exponent that consists of five different scale factors. Each of these factors is thought to exhibit diseconomies of scale in relation to effort. Process Maturity is one of the five scale factors and its rating is based on the SW-CMM. The COCOMO II Post-Architecture model is still undergoing calibration and refinement. This research into Process Maturity's effect on effort is based on the data collected for COCOMO II research. The COCOMO II cost drivers (or predictor variables) are described in section 4.5 on page 37.

Chapter 4

RESEARCH QUESTION AND APPROACH

4.1 The Problem

The Software Capability Maturity Model (SW-CMM) is a specification of what should be in software processes. It does not describe how they should be done nor when they should be performed. The SW-CMM addresses management issues. It discusses the process elements and activities involved in the management of software. It can be used as a roadmap for improving software processes. It can be used as a set of criteria for evaluation of software processes. The SW-CMM is not a quick fix for a project in trouble.

While the SW-CMM is focused on addressing software management issues, it does not address other important areas that affect software development productivity. These areas include development methodologies, technologies, standards, and the need for qualified people; the latter is addressed in a separate People CMM [Curtis 1995]. Other issues not addressed in the current Software CMM are criteria for effective risk management, reuse guidelines, product-line development, and component brokerage, although these are candidates in the current draft of Software CMM version 2.0 [Paulk 1997].

The SW-CMM does not address the need for incentives or career paths that reward the creation and following of successful management processes. Upward career paths frequently are made on short term gains instead of long term investments. Improving processes take time because of the required change in corporate culture and in daily practices. Being a champion of process improvement may not mean advancement or recognition in the organization.

There is a need for a clearer assessment of Process Maturity effects on software development productivity. The case studies show that there are many benefits to improving Process Maturity. However Process Maturity as specified by the SW-CMM does not address all areas that affect productivity on a software development project. The conclusions in the case studies used different assessment approaches, none of which attempt to separate out individual factors that affected productivity. Even with this incomplete analysis, the indication is that increasing maturity levels has generally positive effects.

Many of the case studies describe the benefits of Software Process Maturity in terms of productivity, a controversial measurement. Boehm defines productivity as the ratio of the outputs produced by the process to the inputs consumed by the process, Equation 35 [Boehm, 1987, p. 44]. The difficulty in using this ratio is the controversy of what constitutes the inputs and outputs of the software development process. Outputs can include specification documents, interface documents, design documents, test documents, source code listings, development plans, test cases with data, and user’s manuals. One of the out-

puts, source code listings, is criticized as being ill defined, i.e. what is a line of code, and non-uniform counting, i.e. different lines of code counts produce the same functionality when using different high order languages. Inputs required to produce these outputs are labor, tools, training, computers, facilities. Depending on when in the lifecycle the measurement of inputs begin, additional inputs are specification documents, test documents, and interface documents.

$$\text{Productivity} = \frac{\text{Outputs produced by the process}}{\text{Inputs consumed by the process}} \qquad \text{Equation 35}$$

Instead of productivity, *this research examines Process Maturity's effect on effort,* which is a fundamental component of productivity. However there are many factors that affect the measurement of effort. Effort on a development project consists of developers, project managers, support personnel in specialties such as system administration, configuration management, or quality assurance, and administrative personnel. Factors that affect effort on a development project come from the areas of product characteristics, project management, target platform, and development team qualifications. These will be discussed later. The approach to addressing these concerns is to measure effort spent on a software development project consistently and to measure the factors that influence effort.

4.2 Research Question

My hypothesis is that *increasing the level of Software Process Maturity decreases the amount of software development effort* required to produce a software product; a positive contribution. Case studies have reported [Broadman and Johnson 1995, Butler 1995, Dion 1993, Herbsleb et al. 1997, Humphrey et al. 1991, McGibbon 1996, Springsteen et al. 1992, Wohlwend and Rosenbaum 1994] an increase in productivity resulting from a mix of process-related improvements, e.g. a reduction in rework and a reduction in "re-inventing the wheel," and non-process improvements such as reuse, tools, and personnel.

It is reported in [Dion 1993, Herbsleb et al. 1994] that increasing Process Maturity resulted in a reduction of *rework* which causes a net reduction in effort. The following is a list factors that can cause rework:

- Changing requirements
- Not satisfying requirements
- Unresolved risks
- Poor planning
- Lack of coordination between a development group and/or another development group, customers, users, subcontractors
- Uncoordinated changes in the software product
- Incorrect sequence of work activities (poorly defined software process)
- Poor workmanship in requirements analysis, product design, coding and testing
- Lack of a defect prevention process (detection, feedback, and correction)

In assessing KPA effects on effort, this dissertation includes an analysis organized by software development stage (which may represent a phase in a waterfall model, or a cyclically revisited activity in a spiral model). Appendix A presents the major effort effects

by software lifecycle stage for each KPA. The primary conclusion is that the KPAs' primary contribution to effort reduction is via reduction of rework.

The quantification portion of this dissertation uses a different approach than those found in the case studies. The approach collects and analyzes data to quantify factors that affect software development effort, including Process Maturity. This quantification will determine the magnitude of the effect of Process Maturity on effort and show the quantified relationship between Process Maturity and other factors. This result will be a clear assessment of Process Maturity's effect on effort by separating it from the other factors that influence effort.

A mathematical model is used to quantify the influences that different factors have on development effort. The model's output is the predicted effort required to develop a software product. The position of the factors and their associated coefficients and exponents in the model provide a bases for understanding the effect that one factor has on the model output, effort. The model also makes explicit a factor's relative degree of influence among the other factors in the model.

4.3 The Research Model

After reviewing the existing effort estimation models there are several requirements an estimation model needs to address to support this research:

- The model must support the non-linear relationship between effort and size. The economy/diseconomy of scale relationship has been shown to exist in studies, Table 2.
- The model must use Process Maturity as an input. This will show if Process Maturity can be quantified and it will show the significance of Process Maturity in explaining the variation in effort.
- The model must be accurate. Sufficient accuracy will verify the model form and coefficients as representative of the real world.
- The model must be explainable. The effect on effort of varying each model parameter must be understandable.
- The model should use only enough factors such that the variation in effort is explained and each factor is significant.
- The model should use factors that are independent of each other but related to effort. This prevents double counting and makes the model stable.
- It must be possible to numerically calibrate the factors in the model using multiple regression analysis. This type of analysis also examines interrelationships among the independent variables, reports unexplained variance, provides a goodness of fit of the model to data, and reports the significance that each independent variable has in predicting effort. With the latter analysis, weak independent variables can be removed from the model, thereby permitting the full strength of the remaining variables to be determined.

A Research Model is proposed based on the Log-Log model discussed in section 2.3.1 on page 13. This model will have a coefficient for Process Maturity that quantifies the effect on effort. The predictor variable for Process Maturity is labeled PMAT and its expo-

nent is labeled B_{PMAT}. The set of coefficients for all influencing factors identify which factors are the most influential on determining development effort required to produce a software product. The set of coefficients can also be used to understand the relative strengths between factors from the point of view of influencing effort.

A model is proposed that is non-linear but that can be transformed into a linear model thus permitting the use of linear regression techniques. This model is based on the econometric Log-Log model and is labeled in this dissertation as the Research Model [Griffiths et al. 1993, pp. 258,277].

$$\hat{Y} = A \cdot X_1^{B_1} \cdot X_2^{B_2} \cdot \ldots \cdot X_k^{B_k} \qquad \text{Equation 36}$$

The Research Model in Equation 36 can be transformed into a linear model by taking the logarithm of both sides, Equation 37. This technique was first demonstrated on cost models in 1986 as a suggestion for calibration of COCOMO [Gulezian 1986]. It is also use in the field of Econometrics [Griffiths et al. 1993].

$$\ln(\hat{Y}) = B_0 + B_1 \ln(X_1) + B_2 \ln(X_2) + \ldots + B_k \ln(X_k) \qquad \text{Equation 37}$$

where B_0 is ln(A).

In addition to the Research Model the COCOMO II model is used with the same data to compare and contrast results [Boehm et al. 1995]. The mathematical form of the COCOMO II model is different (see section 3.2.12 on page 32) than the form of the Research Model. Using two models provides the opportunity to consider PMAT's effectiveness differently.

4.4 Hypothesis Testing

The focus of this research is deriving the value for the exponent for the PMAT predictor variable in the Research Model. If Process Maturity affects software development effort, then the coefficient for PMAT should not be zero. Examining the Research Model above, a B_{PMAT} that is zero would make PMAT equivalent to one. A non-zero coefficient would show that PMAT affects effort. The null hypothesis and the alternative hypothesis are stated as follows:

$$H_0: B_{PMAT} = 0$$
$$H_1: B_{PMAT} \neq 0 \qquad \text{Equation 38}$$

The objective of this research is to test the null hypothesis, at the 95% confidence level, to determine whether PMAT does affect effort even after including the effects of other major contributing factors.

In reality though, B_{PMAT}, is not known. It can only be estimated using b_{PMAT}. To successfully conclude that PMAT does affect effort the estimated coefficient, b_{PMAT}, must not be equal to zero. Because b_{PMAT} is an estimate of B_{PMAT}, a test must be performed to

check if zero is within the b_{PMAT} estimation interval. This is called a t-test which checks for Type I errors in hypothesis testing. This was explained in section 2.3.2 on page 15.

The hypothesis will be tested using the t-test. Since B_{PMAT} is not known, the value from the null hypothesis is used instead, $B_{PMAT} = 0$. The t-value acts as a measure of the signal to noise ratio.

$$\text{t-value}_{PMAT} = \frac{b_{PMAT} - B_{PMAT}}{\text{est s.e.}(b_{PMAT})} \qquad \text{Equation 39}$$

4.5 Candidate Predictor Variables

Most analyses identify four areas that influence software development effort. Predictor variables that represent four influential areas are used as inputs into the Research Model. These predictor variables are also in the COCOMO II cost model [Boehm et al., 1995] and they are regrouped into the four areas in Figure 8.

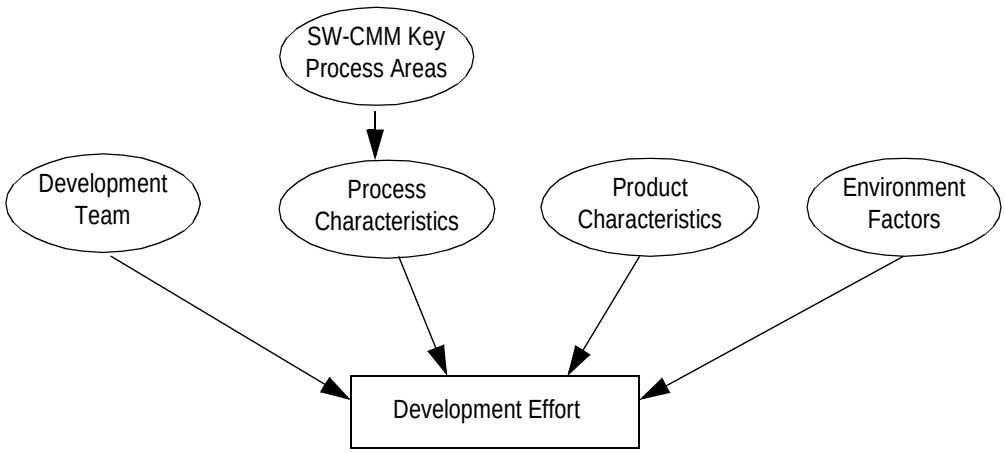


Figure 8. Effort Influencing Areas

The next four subsections are a list of COCOMO II predictor variables that support the four areas shown in Figure 8.

4.5.1 Product Characteristics

The Product characteristics can have a large impact on effort. Product characteristics include size, amount of required software reuse, required reliability, complexity, stor-

age and time constraints, and the stability of the underlying infrastructure on which the software relies.

Table 6. Product-related Predictor Variables

Predictor Variable	Symbol	Description
Size	KSLOC2	Software size is measured in thousands of source lines of code adjusted for reuse and breakage. See section 4.6 on page 42 for an explanation of KSLOC2.
Precedentedness of the application	PREC	This is the extra effort needed because an organization does not understand the software product objectives and has no experience in working on related software systems.
Architecture and risk resolution	RESL	This is the extra effort required because of incompletely specified high-level design or unresolved and unmanaged risks.
Required software reliability	RELY	The measure of the assurance that the software will perform its intended function over a period of time.
Database Size	DATA	Database size attempts to capture the effects of large data requirements on product development.
Product complexity	CPLX	Complexity captures the difficulty of the product development in five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations
Required reuse	RUSE	This accounts for the extra effort needed to build components intended for reuse on the current or future projects.
Documentation match to life-cycle needs	DOCU	This accounts for the extra effort needed to produce documentation that is excessive for the life-cycle of the software
Execution time constraint	TIME	Execution time is a measure of the constraint imposed on the software product to execute within a percentage of the processor's total capacity.
Main storage constraint	STOR	Main Storage Constraint rates the constraint imposed on a software product to fit within a limited storage area.

4.5.2 Development Process

The Development Process directs the activities of the developers, quality assurance personnel, and project management. Activities include SW-CMM-oriented practices such as requirements management, product design, coding, unit testing, integration and test, configuration management, quality assurance, and peer reviews. Although the SW-CMM

specifies a progression on KPAs to attain higher maturity levels, organizations may practice some of the KPAs in all of the levels.

Table 7. Process-related Predictor Variables

Predictor Variable	Symbol	Description
Process Maturity	PMAT	It is the measure of the maturity level of a project's software process. It is either a rating selected from one of six choices (CMM 1 Lower, CMM 1 Upper, CMM 2, CMM 3, CMM 4, CMM 5) OR it is the average of 18 KPA ratings used to assess a process's maturity (Boehm et al. 1995, p.79). See section 4.6.2 on page 46 for an explanation.
Requirements Management	KPA1	Management of requirements allocated to software to resolve issues before they are incorporated into the software project [Paulk et al. 1995a, p.126].
Software Project Planning	KPA2	Developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work [Paulk et al. 1995a, p.133].
Software Project Tracking and Oversight	KPA3	Tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and adjusting these plans based on the actual accomplishments and results [Paulk et al. 1995a, p.148].
Software Subcontract Management	KPA4	Selecting a software subcontractor, establishing commitments with the subcontractor, and tracking and reviewing the subcontractor's performance and results [Paulk et al. 1995a, p.159].
Software Quality Assurance	KPA5	Reviewing and auditing the software products and activities to verify that they comply with the applicable procedures and standards and providing the software project and other appropriate managers with the results of these reviews and audits [Paulk et al. 1995a, p.171].
Software Configuration Management	KPA6	Identifying the configuration of selected software work products at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software life cycle [Paulk et al. 1995a, p.180].
Organization Process Focus	KPA7	Developing and maintaining an understanding of the organization's and projects' software processes and coordinating the activities to assess, develop, maintain, and improve these processes [Paulk et al. 1995a, p.194].
Organization Process Definition	KAP8	Develop and maintain a usable set of software process assets that improve process performance across the projects and provide a basis for cumulative, long- term benefits to the organization [Paulk et al. 1995a, p.202].
Training Program	KPA9	Identifying the training needed by the organization, projects, and individuals, then developing or procuring training to address the identified needs [Paulk et al. 213].

Table 7. Process-related Predictor Variables

Integrated Software Management	KPA10	Integrate the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets [Paulk et al. 1995a, p.223].
Software Product Engineering	KAP11	Integrates all the software engineering activities; such as analyzing the system requirements allocated to software, developing the software architecture, designing the software, implementing the software in the code, and testing the software to verify that it satisfies the specified requirements; to produce and support correct, consistent software products effectively and efficiently [Paulk et al. 1995a, p.241].
Intergroup Coordination	KPA12	Participation with other project engineering groups to address system-level requirements, objectives, process, and issues. Participation in establishing the system-level requirements, objectives, and plans by working with the customer and end users, as appropriate [Paulk et al. 1995a, p.261].
Peer Reviews	KPA13	Methodical examination of software work products by the producers' peers to identify defects and areas where changes are needed [Paulk et al. 1995a, p.270].
Quantitative Process Management	KPA14	Taking measurements of the process performance, analyzing these measurements, and making adjustments to maintain process performance within acceptable limits [Paulk et al. 1995a, p.278].
Software Quality Management	KPA15	Defining quality goals for the software products, establishing plans to achieve these goals, and monitoring and adjusting the software plans, software work products, activities, and quality goals to satisfy the needs and desires of the customer and end user [Paulk et al. 1995a, p.292].
Defect Prevention	KPA16	Analyzing defects that were encountered in the past and taking specific actions to prevent the occurrence of those types of defects in the future [Paulk et al. 1995a, p.306].
Technology Change Management	KPA17	Identifying, selecting, and evaluating new technologies, and incorporating effective technologies into the organization [Paulk et al. 1995a, p.319].
Process Change Management	KPA18	Defining process improvement goals and, with senior management sponsorship, proactively and systematically identifying, evaluating, and implementing improvements to the organization's standard software process and the projects' defined software processes on a continuous basis [Paulk et al. 1995a, p.330].

4.5.3 Development Team

The Development Team affects effort due to its capability, teamwork, experience, continuity, and cohesiveness.

Table 8. Development Team-related Predictor Variables

Predictor Variable	Symbol	Description
Analyst capability	ACAP	Analyst capability rates the personnel that work on requirements, high level design, and detailed design.
Programmer capability	PCAP	Programmer capability rates the project team's ability, efficiency / thoroughness, and ability to communicate.
Applications experience	AEXP	Application experience is a rating dependent on the level of applications experience of the project team.
Platform experience	PEXP	Platform experience rates the understanding of using more powerful platforms such as workstations, graphical user interfaces, databases, networking, and distributed middleware.
Language and tool experience	LTEX	Language and Tool Experience is a measure of the level of programming language and software tool experience of the project team.
Personnel continuity	PCON	Personnel continuity is a rating scaled for the project's annual personnel turnover.
Development team cohesion	TEAM	This is the extra effort required on software projects whose developers, customers, and users have difficulty in coordinating their activities.

4.5.4 Environment Factors

The Environment factors that affect effort are technology insertion (such software engineering methods and tools), facilities, and work conditions (such as multi-site development or development schedule compression).

Table 9. Environmental-related Predictor Variables

Predictor Variable	Symbol	Description
Development Flexibility	FLEX	This is the required conformity to development standards and constraints such as rigid schedules or performance requirements. It accounts for the extra effort needed to follow rigid and inflexible software development standards and constraints.
Use of software tools	TOOL	Use of Software Tools rates the use of tools in making the software development more efficient.
Multi-site development	SITE	This accounts for the extra effort needed to coordinate and integrate development activities that are not co-located and do not have access to wideband electronic communication facilities.

Table 9. Environmental-related Predictor Variables

Platform volatility	PVOL	Platform Volatility is a rating of the frequency of change in the complex of hardware and software that the product calls upon to do its work.
Required development schedule	SCED	Required Development Schedule measures the schedule constraint imposed on the project team developing the software, e.g. schedule compression.

4.6 Collecting Data

Data is collected on the product size, the actual effort expended on the project, and the predictor variables. Sizing data consists of a count of new lines of code developed, lines of code adapted from previous projects, and the amount of code breakage, i.e. code that was developed but not used. Size is computed as in Equation 40.

$$KSLOC2 = \frac{\left(NSLOC \cdot \frac{(BRAK + 100)}{100}\right) + (ASLOC \cdot 0.20)}{1000} \qquad \text{Equation 40}$$

The sizing equation incorporates both newly developed lines of code, NSLOC, and code that is adapted from other software developments, ASLOC. The new code is adjusted for breakage, BRAK, which is additional code written but not used. A percentage of code size from adapted code modules is used to represent an equivalent size of newly developed code. The Manager's Handbook for Software Development from the NASA Software Engineering Laboratory uses 20% of the adapted code size for computing equivalent lines of newly developed code with the restriction that not more than 25% of the adapted code module has been modified [SEL 1990, Table 3-8]. KSLOC2 contains a count of both new and equivalent lines of code.

The KSLOC2 sizing formula is different than that specified for the COCOMO II model [Boehm et al. 1995, pp. 60-62]. The COCOMO II sizing formula has a more sophisticated approach to counting adapted code. It accounts for the non-linear percentage of effort required to modify adapted code. It accounts for the percentage change in module design and changes in the integration required for the software. It also accounts for the effort saving effect of structured, documented code or people on the project that have worked previously on the adapted code. The 20% used in computing KSLOC2, Equation 40, is a gross approximation of these effects. It is used to enable use of some data points for which the complete set of COCOMO II reuse parameters are not available.

Effort is measured in Person Months. A person month is 152 hours. It includes the software developer's time, project management time, administrative support time, and project support personnel time, e.g. configuration management and quality assurance. The period measured on a project was from completion of requirements analysis to the end of integration and test.

4.6.1 Collecting Data on Predictors

The data collection form in Appendix B was used to collect data. This is the same data collection form and definitions used to collect data for the COCOMO II model. The rating names have been changed from COCOMO's Very Low, Low, Nominal, High, Very High, and Extra High to R1, R2, R3, R4, R5, and R6 in this dissertation. In COCOMO, the nominal rating is predefined. For research purposes the median rating for each predictor from the collected data is designated as nominal. The median values for all predictors are given in section C.1.1 on page 105. The nominal rating for this research is assigned the value of 1.0 as in the COCOMO II model.

Each predictor variable can have up to six ratings, R1 through R6. Some of the predictors have less than six ratings because of the scale definition. For instance TIME and STOR only have four ratings. A rating less than R3 would not impact effort therefore no ratings are assigned. Table 10 shows the ratings and the definitions for the rating for each predictor variable.

Table 10. Rating Criteria

	R1	**R2**	**R3**	**R4**	**R5**	**R6**
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
RESL	little (20%)	some (40%)	often (60%)	generally (75%	mostly (90%)	full (100%)
RELY	slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
DATA		DB bytes/ Pgm SLOC < 10	10 (D/P < 100	100 (D/P < 1000	D/P (1000	
CPLX	see Table 11					
RUSE		none	across project	across program	across product line	across multiple product lines
DOCU	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	
TIME			(50% use of available execution time	70%	85%	95%
STOR			(50% use of available storage	70%	85%	95%

Table 10. Rating Criteria

	R1	R2	R3	R4	R5	R6
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative interactions	highly cooperative	seamless interactions
ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCON	48% / year	24% / year	12% / year	6% / year	3% / year	
AEXP	≤ 2 months	6 months	1 year	3 years	6 years	
PEXP	≤ 2 months	6 months	1 year	3 years	6 year	
LTEX	≤ 2 months	6 months	1 year	3 years	6 year	
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
TOOL	edit, code, debug	simple, front-end, back-end CASE, little integration	basic lifecycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	
PVOL		major change every 12 mo.; minor change every 1 mo.	major: 6 mo.; minor: 2 wk.	major: 2 mo.; minor: 1 wk.	major: 2 wk.; minor: 2 days	
SITE: Collocation	International	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metro. area	Same building or complex	Fully collocated
SITE: Communications	Some phone, mail	Individual phone, FAX	Narrowband email	Wideband electronic communication.	Wideband elect. comm, occasional video conf.	Interactive multimedia
SCED	75% of nominal	85%	100%	130%	160%	
PMAT	CMM Lvl 1 (Lower)	CMM Lvl 1 (Upper)	CMM Lvl 2	CMM Lvl 3	CMM Lvl 4	CMM Lvl 5
	or an average of KPA Goal compliance, see Equation 41					

Product complexity is an average of five different measures. Table 11 shows the different measures and the associated ratings.

Table 11. Complexity Ratings

	Control Operations	**Computational Operations**	**Device-dependent Operations**	**Data Management Operations**	**User Interface Management Operations**
R1	Straight-line code with a few non-nested structured programming operators: DOs, CASEs, IF-THENELSEs. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions: e.g., *A=B+C*(D-E)*	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTS-DB queries, updates.	Simple input forms, report generators.
R2	Straightforward nesting of structured programming operators. Mostly simple predicates	Evaluation of moderate-level expressions: e.g., *D=SQRT(B**2-4.*A*C)*	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/ PUT level.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Use of simple graphic user interface (GUI) builders.
R3	Mostly simple nesting. Some intermodule control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.
R4	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, roundoff concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlap.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O, multimedia.

Table 11. Complexity Ratings

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
R5	Reentrant and recursive coding. Fixed-priority interrupt handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
R6	Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Distributed hard real-time control.	Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data. Complex parallelization.	Device timing-dependent coding, micro-programmed operations. Performance-critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality.

4.6.2 Collecting Process Maturity Data

Data on Process Maturity is collected at the project level. While an organization may be rated at a specific SW-CMM level, the respondents were encouraged to answer all of the KPA questions considering *what actually happened on the project.* Respondents had two ways to respond to rating Process Maturity. The first way was by selecting an overall maturity level based either on an organized evaluation or subjective judgment, Figure 9.

The selection for SW-CMM Level 1 (lower half) is for organizations that rely on "heroes" to get the job done. There is no focus on processes or documenting lessons learned. The SW-CMM Level 1 (upper half) is for organizations that have implemented most of the KPA goals that would satisfy SW-CMM Level 2. SW-CMM Level 2 is for organizations that have processes in place that will permit repeated successes on projects. These organizations manage requirements, plan and track projects and employ configuration management at the project level. SW-CMM Level 3 is for organizations that maintain organization level processes used to guide individual project processes. The organization learns by continually improving processes to reflect lessons learned. SW-CMM Level 4 and 5 are for organizations that have been assessed as meeting Level 4 or Level 5 KPA goals.

The second way of rating Process Maturity is to rate the percentage of compliance for each set of KPA goals. The data collection form in Appendix B, shows each KPA and its goals. The goals for each KPA are considered and a rating is selected that reflects the percentage of compliance by the project. Table 12 shows the KPA rating weights.

Overall Maturity Level

❒ CMM Level 1 (lower half)
❒ CMM Level 1 (upper half)
❒ CMM Level 2
❒ CMM Level 3
❒ CMM Level 4
❒ CMM Level 5

Figure 9. Maturity Level

Table 12. KPA Rating Weights

Rating	Description	Weight
Almost Always	When the goals are consistently achieved and are well established in standard operating procedures (over 90% of the time)	100
Frequently	When the goals are achieved relatively often, but sometimes are omitted under difficult circumstances (about 60 to 90% of the time)	75
About Half	When the goals are achieved about half of the time (about 40 to 60% of the time)	50
Occasionally	When the goals are sometimes achieved, but less often (about 10 to 40% of the time)	25
Rarely If Ever	When the goals are rarely if ever achieved (less than 10% of the time)	1
Does Not Apply	When you have the required knowledge about your project or organization and the KPA, but you feel the KPA does not apply to your circumstances	0
Don't Know	When you are uncertain about how to respond for the KPA. After the level of KPA compliance is determined each compliance level is weighted and a PMAT factor is calculated, as in Equation 13. Initially, all KPAs will be equally weighted	0

PMAT is computed as the average of all rated KPAs (Does Not Apply and Don't Know are not counted which sometimes makes n less than 18).

$$PMAT = \left(\sum_{i=1}^{n} \frac{KPA\ \%}{100} \right) \cdot \frac{1}{n} \qquad \text{Equation 41}$$

The KPA data is collected at the project level. This level of information is desired so that the effects of Process Maturity can be assessed at the project level. Table 13 shows an example page from the KPA entry form in Appendix B. There are eighteen KPAs and each has five ratings for a total of ninety possible selections.

Table 13Example of KPA Collection

Key Process Area	Goals of each KPA	Almost Always	Frequently	About Half	Occasionally	Rarely If Ever	Does Not Apply	Don't Know
Requirements Management: involves establishing and maintaining an agreement with the customer on the requirements for the software project.	System requirements allocated to software are controlled to establish a baseline for software engineering and management use. Software plans, products, and activities are kept consistent with the system requirements allocated to software.	☐	☐	☐	☐	☐	☐	☐
Software Project Planning: establishes reasonable plans for performing the software engineering activities and for managing the software project.	Software estimates are documented for use in planning and tracking the software project. Software project activities and commitments are planned and documented. Affected groups and individuals agree to their commitments related to the software project.	☐	☐	☐	☐	☐	☐	☐
Software Project Tracking and Oversight: provides adequate visibility into actual progress so that management can take corrective actions when the software project's performance deviates significantly from the software plans.	Actual results and performances are tracked against the software plans. Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans. Changes to software commitments are agreed to by the affected groups and individuals.	☐	☐	☐	☐	☐	☐	☐
Software Quality Assurance: provides management with appropriate visibility into the process being used by the software project and of the products being built.	Software quality assurance activities are planned. Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively. Affected groups and individuals are informed of software quality assurance activities and results. Noncompliance issues that cannot be resolved within the software project are addressed by senior management.	☐	☐	☐	☐	☐	☐	☐

4.7 Approach to Quantification of Qualitative Data

4.7.1 Assigning Values to Ratings

To convert a predictor rating, an ordinal value, to a quantified value for use in the Research Model a monotonic sequence of numbers that pass through the median rating, 1.0, are assigned to each rating. The sequence of numbers assigned to the PMAT rating should be decreasing from R1 to R6 and should use the number one as the median value (see section 5.3 on page 52 for the actual values assigned). The sequence decreases to test the hypothesis that as higher levels of Process Maturity are attained (moving towards the R6 rating) the software development effort should decrease.

Chapter 5

RESULTS

5.1 Data Description

There are one hundred twelve project observations used in this research. The data is all qualitative except for size and effort. The data is stored symbolically and it is instantiated with a set of predictor values for use in analysis. The values and the rationale for creating them is given in this chapter. The same data set can be instantiated with different value sets depending on the cost model specification. Results of this research used a value set for the Research Model and a different value set for the COCOMO II model. Both sets are described later.

The data came from eighteen sources. These sources covered the Aerospace Industry, Federally Funded Research Centers, Commercial Industry, and Department of Defense supported Industry. The data was on past, completed projects. Much of the data is proprietary and furnished to the University of Southern California under nondisclosure agreements. The data cannot unfortunately be included in this document. However the data is described in Appendix C.

Most of the data came from 1990's projects, although some projects from the 1970's and 1980's are included. Product sizes range from 2.6 to 1264 KSLOC, Figure 10[1]. The KSLOC data has an average of 158, a median of 53 and a standard deviation of 265. Project effort ranges from 6 to 11,400 Person Months, Figure 11. PM data has an average of 830, a median of 180 and a standard deviation of 2001. Process maturity levels cover the full range. The proportion of Levels 3, 4, and 5 projects is higher than the community-wide distributions shown in Figure 2. This is due to the higher emphasis on data collection and analysis at the higher process maturity levels. See Appendix C for the distribution of PMAT and KPA values among the one hundred twelve projects.

While the data sources varied there was selection bias in the data. We were not given data on unfinished or unsuccessful projects nor did any unsuccessful companies contribute data. The data was from successful projects from successful companies. Proof of this is in the fact that these companies were mature enough to practice collecting data and that the project had to finish in order to provide completed data.

The data collected was on the predictor variables and the actual effort expended during the project. Supporting information such as application domain and reuse sizing data

1. In the histogram, a count was added to the bin if a value was equal to or less than the bin marker. For example, there are 10 counts in the 20 bin which means a project's size was equal to or less than 20 KSLOC but greater than 10 KSLOC, the next bin.

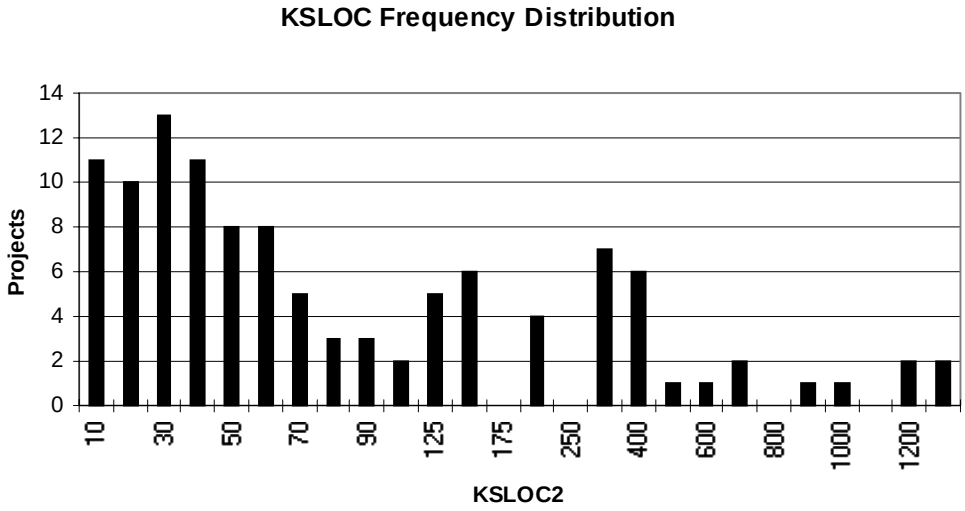


Figure 10. KSLOC Distribution

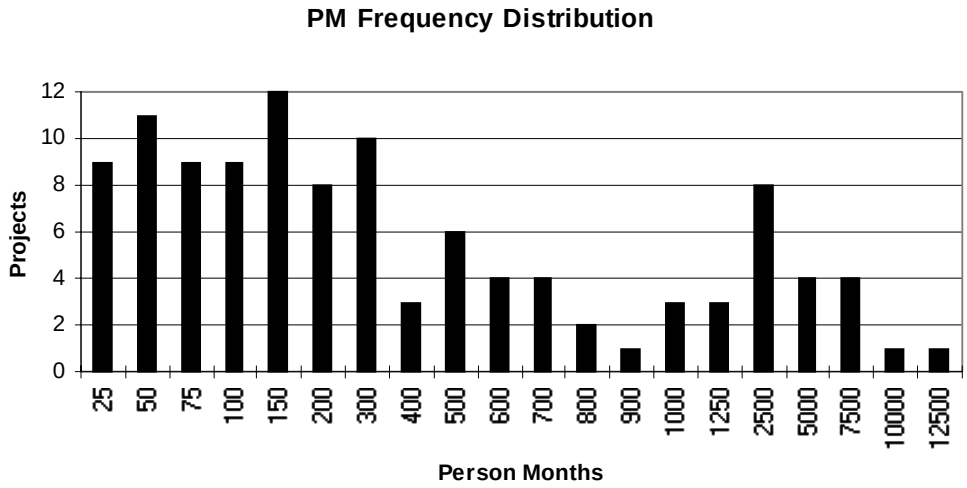


Figure 11. PM Distribution

was collected but it is incomplete. There was no data collected on whether the software technology used on the project matched the application complexity. There was no data collected on whether the processes used to develop the software were mismatched to the type of application domain. While it was requested, uncompensated overtime was not consis-

tently collected. These and other factors, such as the interpretation of qualitative ratings, mean that the data are imprecise. Precise results should not be expected.

5.2 Collinearity Test Results

Appendix C has the results of pairwise correlation for all the Predictor Variables. The correlation results show three sets of correlations. The first is Analyst Capability (ACAP) and Programmer Capability (PCAP) with a correlation of 0.66. Many respondents could not distinguish between the two predictor variables because the development teams did both analysis and programming. These two predictors are combined into a new predictor called Personnel Capability (PERS) using the geometric mean of the two predictors' ratings.

The next pairwise correlation is between Execution Time Constraint (TIME) and Main Storage Constraint (STOR) with a correlation of 0.67. Generally, the target platform on which the software was going to execute was either constrained in both TIME and STOR or neither predictor had any constraints. A satellite is an example of the former platform and a high performance workstation with virtual memory is an example of the latter. These predictors are combined into a new predictor, Resource Constraints (RCON), using the geometric mean of the two predictors' ratings.

The third pairwise correlation is Language and Tool Experience (LTEX) and Platform Experience (PEXP) with a correlation of 0.65. This frequently results from organizations operating on a stable platform, programming language, and toolset. They are not combined as their computed effects were relatively small.

When performing the analysis on the data, the new predictors, PERS and RCON, are used.

Correlation between 3 or more predictor variables was investigated using Principal Components Analysis [Weisberg 1985, pp.186-188]. No group correlations were found.

5.3 PMAT Quantification

Process Maturity is measured either by selecting one of six ratings or by averaging the level of compliance with each set of KPA goals (this was explained in section 4.6.2 on page 46). From one of these two selection criterion, PMAT is rated with one of six possible ratings, R1 through R6.

All of the ratings in the data for PMAT were examined to find the median rating. The median value for PMAT was the R3 rating, i.e. the data showed that R3 was the median of responses. The median rating was given a value of 1.0. The rating on either side of the median was given values that differed from 1.0 by 10%.

Considering the elasticity interpretation for the Research Model (see section 2.3.1 on page 13), a uniform 10% change in value between ratings was chosen to simplify interpretation of the model. These values are used to initialize the statistical analysis process. Uniform changes of 5% or 20% would produce essentially the same end results. The scale and the corresponding ratings are shown in Figure 12.

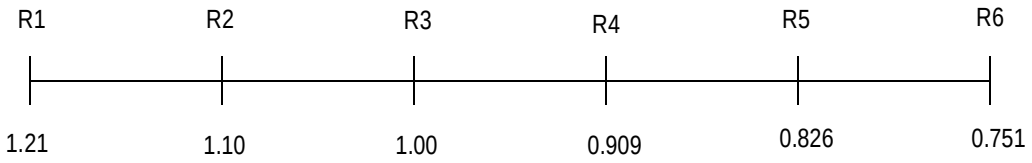


Figure 12. PMAT Rating Value Range

5.4 Research Model Predictor Values

The approach taken for all qualitative predictor variables is that the frequency of responses for each rating determines the median. Appendix C has the distribution for all of the predictor variables. The median for most predictors was between R3 and R4. If the median falls on 3.5, between R3 and R4, then the ratings values for R1 through R6 are interpolated.

Table 14 contains the values assigned to the symbolic data for one hundred twelve observations. The values are based on a 10% interval as explained above. There are blanks in the table which represent invalid ratings.

Table 14. Research Model Predictor Values

Cost Driver	R1	R2	R3	R4	R5	R6	PR
PREC	1.27	1.16	1.05	0.954	0.867	0.788	1.61
FLEX	1.21	1.1	1	0.909	0.826	0.751	1.61
RESL	1.27	1.16	1.05	0.954	0.867	0.788	1.61
TEAM	1.27	1.16	1.05	0.954	0.867	0.788	1.61
PMAT	1.21	1.1	1	0.909	0.826	0.751	1.61
RELY	0.751	0.826	0.909	1	1.1		1.46
DATA		0.867	0.954	1.05	1.16		1.33
CPLX	0.788	0.867	0.954	1.05	1.16	1.27	1.61
RUSE		0.909	1	1.1	1.21	1.33	1.46
DOCU	0.826	0.909	1	1.1	1.21		1.46
TIME			0.954	1.05	1.16	1.27	1.33
STOR			0.954	1.05	1.16	1.27	1.33
PVOL		0.909	1	1.1	1.21		1.33

Table 14. Research Model Predictor Values

ACAP	1.33	1.21	1.1	1	0.909		1.46
PCAP	1.33	1.21	1.1	1	0.909		1.46
PCON	1.27	1.16	1.05	0.954	0.867		1.46
AEXP	1.33	1.21	1.1	1	0.909		1.46
PEXP	1.27	1.16	1.05	0.954	0.867		1.46
LTEX	1.27	1.16	1.05	0.954	0.867		1.46
TOOL	1.1	1	0.909	0.826	0.751		1.46
SITE	1.4	1.27	1.16	1.05	0.954	0.867	1.61
SCED	1.21	1.1	1	1.1	1.21		1.46

The PR column is for the productivity range. This range is found by taking the difference between each rating and raising it to the number of intervals, e.g., 6 ratings represent 5 intervals, $(1.1)^5 = 1.61$.

5.5 Research Model Results

5.5.1 The Full Model

The results of the statistical analysis of the full Research Model are given in section D.1 on page 120. The R^2 value is high (0.936) and a number of t-values are high, but the Research Model with all of the predictors in it has some problems. Estimated negative coefficients produce results counter to the understanding of what the model is attempting to predict. A negative estimate means that as the ratings for a predictor (e.g., Applications Experience, AEXP) get more difficult less effort is required; see Equation 42. The negative estimates are generally a result of predictors either having all responses fall within one or two ratings (weak dispersion) or being marked as "I don't know" which were given rating values of 1.0. These negative predictors all have low, statistically insignificant t-values, and are removed from the Research Model.

$$\begin{aligned} PM = {} & 2.22 \cdot KSLOC2^{1.05} \cdot PMAT^{1.49} \cdot PREC^{0.47} \cdot RESL^{-0.23} \\ & \cdot RELY^{0.26} \cdot DATA^{0.88} \cdot CPLX^{1.45} \cdot RUSE^{-0.26} \cdot DOCU^{0.32} \\ & \cdot RCON^{3.55} \cdot PERS^{2.64} \cdot AEXP^{-0.39} \cdot PEXP^{0.86} \cdot LTEX^{-0.47} \\ & \cdot PCON^{0.22} \cdot TEAM^{0.48} \cdot FLEX^{0.17} \cdot TOOL^{-0.26} \cdot SITE^{1.32} \\ & \cdot PVOL^{0.84} \cdot SCED^{3.1} \end{aligned} \qquad \text{Equation 42}$$

The t-value for PMAT is 2.2 in Equation 42 which exceeds the significance threshold of 1.96. The exponent value of 1.49 indicates that the effect of increasing one PMAT rating level is an effort decrease of about 15% rather than 10%.

5.5.2 Pruning Predictors from the Research Model

The analysis results in Appendix D.1 show that some predictors are not significant, with low t-values. There are several possible reasons for this. First, the reason could be that the predictor is not important in influencing software development effort. Yet the majority of these predictors came from the successful COCOMO 1981 model [Boehm, 1981] where they proved important. Other reasons include weak dispersion and effects of partially correlated variables. The final reason relates to the imprecise nature of software data. There is not enough data to support the estimation of all the predictors in the model at the same time. This is a more likely explanation. A statistician's rule of thumb for the amount of data needed to calibrate a model of this size is four observations for each rating. There are on the average five ratings per predictor, twenty predictors, and four observations needed for each rating. This is four hundred observations required and the data set used in this research had only one hundred twelve. Thus, not all of the variables were likely to be statistically significant.

5.5.3 Reduced Research Model

This model is the result of removing predictors that were insignificant (t-value less than 1.96). All of the remaining predictors are significant. However, the scope of parameter coverage over the four effort influencing areas, product, process, team, and environment, is not broad enough to claim this model accounts for all factors that influence effort (a goal of this research). The model may be strong but alternative data sets could substantially change the estimates of the coefficients consequently making them insignificant. The analysis results are given in section 1.2 on page 122; the resulting estimation equation is:

$$PM = 2.19 \cdot KSLOC2^{1.05} \cdot PMAT^{1.31} \cdot DATA^{0.86} \cdot CPLX^{1.44} \cdot RCON^{3.83} \cdot PERS^{2.38} \cdot TEAM^{0.88} \cdot SITE^{1.52} \cdot PVOL^{1.28} \cdot SCED^{3.09} \quad \text{Equation 43}$$

The PMAT predictor has an estimated coefficient of 1.31 and is significant with a t-value of 2.69. The proportional error of the model is shown in the histogram[2] below.

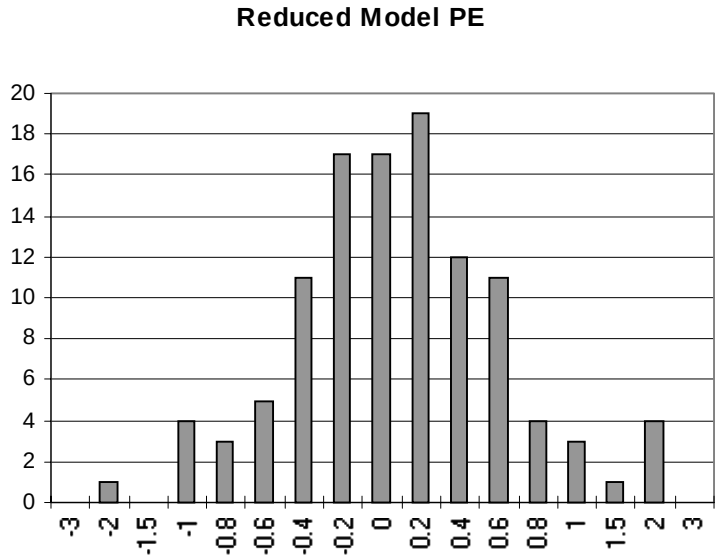


Figure 13. Histogram of Reduced Model PE

5.5.4 Compact Research Model

The Compact Research Model uses parameters that are aggregates of the predictors. The aggregates were created by taking the product of the predictors in the effort-influencing groups Product (PROD), Development Team (DEVT), and Environment (ENVR). These sets of predictors were discussed in section 4.5 on page 37. The analysis results are given in section 1.4 on page 124; the resulting estimation equation is:

$$PM = 2.87 \cdot KSLOC2^{1.03} \cdot PMAT^{2.02} \cdot PROD^{0.65} \cdot DEVT^{0.36} \cdot ENVR^{0.83} \quad \text{Equation 44}$$

2. In the histogram, a count was added to the bin if a proportional error was equal to or less than the bin marker. For example, there are 19 counts in the 0.20 bin which means a project's PE was equal to or less than 0.20 but greater than 0, the next bin. This gives the histogram the appearance of being positively skewed.

Because this model is an aggregation of predictor sets it has the potential to be stable with different data sets. PMAT has an estimate of 2.02 with a t-value of 4.24. The proportional error is shown below.

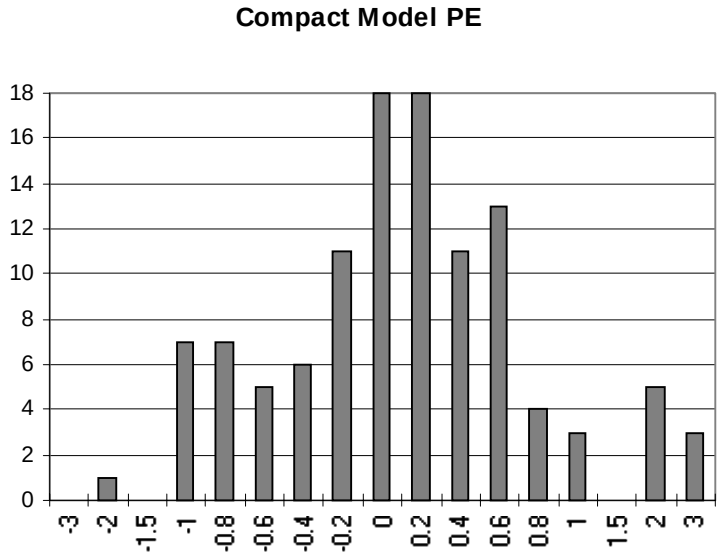


Figure 14. Histogram of Compact Model PE

5.5.5 Small Research Model

The Small Research Model is a roll up of all the predictors, except PMAT, into a composite variable called Effort Multipliers (EM). The advantage of the model is that it should be robust across different data sets. In the process of defining the other Research Models it was observed that as some predictor variables were added or deleted, the coefficients for the remaining predictors in the model changed. This is due to the interaction that is always present among predictor variables, however slight; see Appendix C. When predictors are combined those interactions disappear and the aggregated predictor variable gets a stable estimated coefficient.

The disadvantage of using the Small Research Model is that it is hard to interpret. It is difficult to distinguish what the effect changing one of the predictor variables rolled up into EM will have on effort. Additionally, the relative influence of the predictors in EM cannot be compared to PMAT. PMAT's coefficient estimate is 2.11 with a significant t-value of 4.77, see section 1.6 on page 125. The resulting estimation equation is:

$$PM = 3.0 \cdot KSLOC2^{1.02} \cdot PMAT^{2.11} \cdot EM^{0.63} \qquad \text{Equation 45}$$

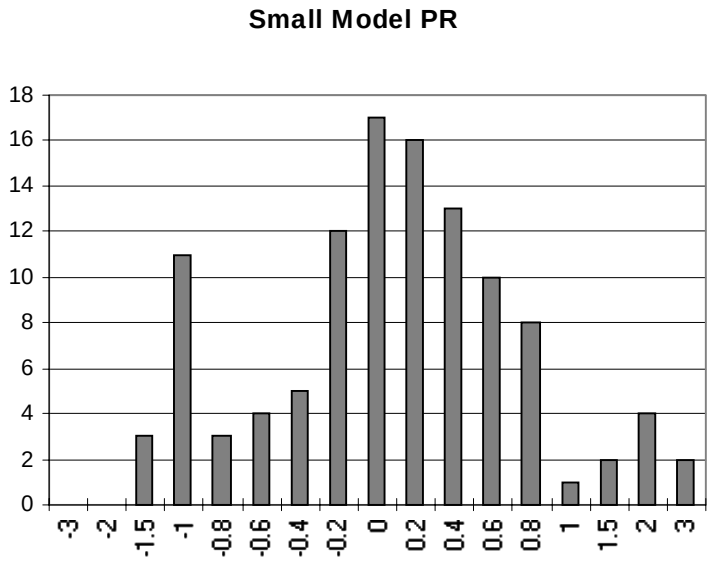


Figure 15. Histogram of Small Model PE

5.5.6 Summary of Research Model Forecast Results

Table 15 shows the model fit to the data. Adj-R^2, est s.e (or standard deviation), and the accuracy of the different model forecasts are discussed in section 2.3.4 on page 17. Recall that the standard deviation for PM is 2001. The models have an estimated standard error between 611 and 769 PM. This is between 31% and 38% of the actual PM standard deviation. The different forms of the Research Model have the model fit and forecast accuracy shown in Table 15. These results will be compared to the accuracy of the forecast tests presented later.

Table 15. Research Model Accuracy

Models	Adj-R^2	est s.e.	PRED(20)	PRED(40)
Full RM	.89	633	32%	60%
Reduced RM	.87	611	32%	58%
Compact RM	.84	692	32%	52%
Small RM	.81	769	30%	52%

5.5.7 Summary of PMAT Results

The estimation interval for PMAT's coefficient is given below for the four Research Models. The interval is depicted in Figure 16. The interval for the Full Research Model is large due to PMAT's high est. s.e. (prediction intervals are explained in section 2.3.1 on page 13). The interval between 1.5 and 2.0 appear within all of the model estimation intervals.

Full RM: $0.20 \leq b_{PMAT} \leq 2.8$

Reduced RM: $0.32 \leq b_{PMAT} \leq 2.3$

Small RM: $1.22 \leq b_{PMAT} \leq 3.0$

Compact RM: $1.05 \leq b_{PMAT} \leq 2.9$

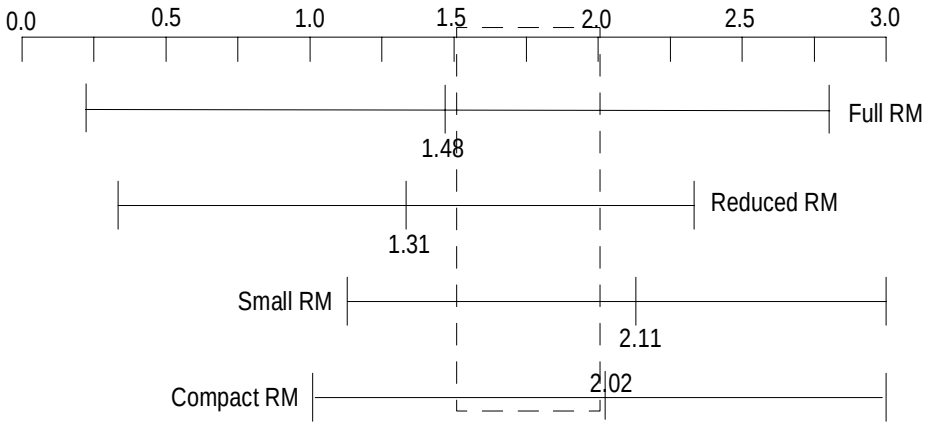


Figure 16. Estimated RM PMAT Interval

Recall that the b values multiplicatively adjust the 10% baseline values assigned in Table 14 with the effect of reducing effort of a one-level change in process maturity. When the coefficient range of 1.5 to 2.0 is applied to PMAT's productivity range (discussed in section 5.4 on page 53), it is stretched from 1.61 to 2.04 for $b_{PMAT} = 1.5$ and to 2.59 for $b_{PMAT} = 2.0$. When these productivity ranges are allocated across the five intervals for the six ratings, a change in one rating level for PMAT becomes $(2.04)^{1/5} = 1.153$ for $b_{PMAT} = 1.5$ and 1.21 for $b_{PMAT} = 2.0$. Thus it is conservative to say that for a one level change or a 10% change in PMAT there is between a 15.3% to 21% decreasing change in software development effort.

5.6 COCOMO II Model

The COCOMO II model has PMAT as an input parameter [Boehm et al. 1995]. The same data set as used on the Research Model can also be used on the COCOMO II model but the symbolic data is instantiated with COCOMO II provisional values instead of the values given in Table 14.

Table 16. COCOMO II Provisional Values

Cost Driver	VL	L	N	H	VH	XH
PREC	0.05	0.04	0.03	0.02	0.01	0
FLEX	0.05	0.04	0.03	0.02	0.01	0
RESL	0.05	0.04	0.03	0.02	0.01	0
TEAM	0.05	0.04	0.03	0.02	0.01	0
PMAT	0.05	0.04	0.03	0.02	0.01	0
RELY	0.75	0.88	1	1.15	1.4	
DATA		0.94	1	1.08	1.16	
CPLX	0.75	0.88	1	1.15	1.3	1.65
RUSE		0.89	1	1.16	1.34	1.56
DOCU	0.85	0.93	1	1.08	1.17	
TIME			1	1.11	1.3	1.66
STOR			1	1.06	1.21	1.56
PVOL		0.87	1	1.15	1.3	
ACAP	1.5	1.22	1	0.83	0.67	
PCAP	1.37	1.16	1	0.87	0.74	
PCON	1.26	1.11	1	0.91	0.83	
AEXP	1.23	1.1	1	0.88	0.8	
PEXP	1.26	1.12	1	0.88	0.8	
LTEX	1.24	1.11	1	0.9	0.82	
TOOL	1.2	1.1	1	0.88	0.75	
SITE	1.24	1.1	1	0.92	0.85	0.79
SCED	1.23	1.08	1	1	1	

5.6.1 Full COCOMO II Model

The one hundred twelve observations were used to estimate coefficients for all of the predictors. The complete results are in section 1.8 on page 126. As with the Research Model there are some estimates that are negative. These values contradict the model scales as defined in Table 16. The parameters are removed in the Reduced COCOMO II model with the same caveat as for the Research Model.

$$\begin{aligned} PM = {} & 2.97 \cdot SIZE^{0.81} \cdot SIZE^{(PREC \cdot 0.9)} \cdot SIZE^{(FLEX \cdot 1.26)} \\ & \cdot SIZE^{(RESL \cdot -0.18)} \cdot SIZE^{(TEAM \cdot 1.6)} \cdot SIZE^{(PMAT \cdot 4.22)} \\ & \cdot RELY^{0.82} \cdot DATA^{0.79} \cdot RUSE^{-0.31} \cdot DOCU^{-0.02} \\ & \cdot CPLX^{1.26} \cdot RCON^{1.71} \cdot PERS^{1.87} \cdot AEXP^{-0.4} \cdot PEXP^{1.27} \\ & \cdot LTEX^{-0.71} \cdot PCON^{-0.14} \cdot TOOL^{-0.14} \cdot SITE^{0.32} \cdot PVOL^{0.46} \\ & \cdot SCED^{2.64} \end{aligned} \qquad \text{Equation 46}$$

The t-value for PMAT is 3.05, which exceeds the significance threshold of 1.96 and exceeds the full RM PMAT's t-value of 2.2. The coefficient for PMAT is 4.22 which strengthens the provisional values assigned to PMAT in Table 16. The strong significance and the large coefficient suggest that PMAT has a diseconomy of scale influence (section 2.2.3 on page 9) on development effort.

5.6.2 Reduced COCOMO II Model

The reduced COCOMO II Model is using the same set of predictors as the Reduced Research Model. The estimates with their t-values are given in section 1.10 on page 128.

$$\begin{aligned} PM = {} & 2.77 \cdot SIZE^{0.89} \cdot SIZE^{(TEAM \cdot 2.64)} \cdot SIZE^{(PMAT \cdot 3.56)} \\ & \cdot DATA^{0.89} \cdot CPLX^{1.4} \cdot RCON^{2.23} \cdot PERS^{1.72} \cdot SITE^{1.13} \\ & \cdot PVOL^{0.92} \cdot SCED^{2.91} \end{aligned} \qquad \text{Equation 47}$$

For the reduced model, PMAT has a very strong estimate, 3.56, and a significant t-value, 3.26. PMAT's effect on effort is strong and significant. However the magnitude of PMAT's influence varies with SIZE. The distribution of KSLOC for the data used in this research is shown in Figure 10.

5.7 Comparison of Results for the Research and COCOMO II Models

Both models show that PMAT is statistically significant. Yet both models treat PMAT differently in that it is specified differently in each equation. It is possible to setup an example and compare the models directly if representative values are chosen for the COCOMO II model inputs. Looking at the distribution of KSLOC in the data used for this research, 30 KSLOC and 150 KSLOC appear to be two representative sizes. When PMAT is set to COCOMO II provisional values, it increases 0.01 for each change in increment in rating. From Figure 17, the value 4.0 can be used as a representative value for PMAT because

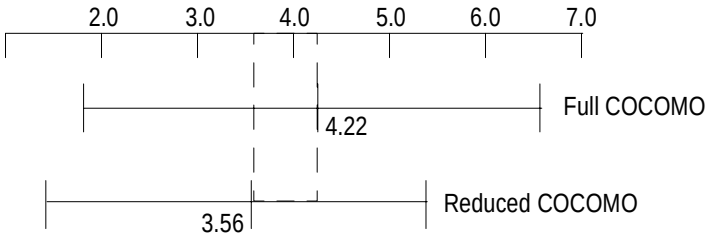


Figure 17. Estimated COCOMO II PMAT Interval

it is within the estimation interval both the Full and Reduced COCOMO II models. Applying this value to PMAT causes it to increase 0.04 for each increment in rating.

Now consider an example in which the project's COCOMO II scale factors yield an exponent of 1.10 (see section 3.2.12 on page 32 for the model description). If PMAT were found to have no influence, this exponent would not change from 1.10 when PMAT's rating was increased one level for a 30 KSLOC project, $(30\ KSLOC)^{1.10} = 42.2$ PM. Using the PMAT adjusted value (0.04 increase for each rating), increasing PMAT's rating by one level changes the exponent from 1.10 to 1.14, yielding $(30\ KSLOC)^{1.14} = 48.3$ PM. This represents a 14.4% increase: 48.3 / 42.2 = 1.144. Using 150 KSLOC yields a 22% increase: $(150)^{1.14} / (150)^{1.10} = 1.22$. This example shows that the six models (four Research and two COCOMO II models) produce similar results for PMAT with the COCOMO II model having a more statistically significant PMAT coefficient.

5.8 Model Forecast Accuracy

A method called cross validation is used to test each model's forecast accuracy [Weisberg 1985, pg.229]. In cross validation the data are split into two groups. The first group is used to estimate the model coefficients, called the calibration set. Seventy-five percent of the observations (84/112 observations) are used in the calibration set. The second group is used for validation of the model, called the validation set. Twenty-five percent of the observations (28/112 observations) comprise this group. The Adj-R^2, estimated Standard Deviation, SD, and the prediction level at 25% and 40%, PRED(25,40), should be similar between the calibration set and validation set.

Table 17. Calibration Set Results

Models	Adj-R^2	SD	PRED(20)	PRED(40)
Full RM	0.79	665	33%	68%
Reduced RM	0.83	592	36%	58%

Table 17. Calibration Set Results

Compact RM	0.77	697	37%	51%
Small RM	0.76	720	36%	56%
Full COCOMO II	0.79	658	36%	61%
Reduced COCOMO II	0.80	674	29%	56%

Table 18. Validation Set Results

Models	Adj-R^2	SD	PRED(20)	PRED(40)
Full RM	0.87	689	18%	36%
Reduced RM	0.89	523	14%	46%
Compact RM	0.89	545	11%	37%
Small RM	0.85	604	14%	42%
Full COCOMO II	0.89	526	25%	46%
Reduced COCOMO II	0.90	490	25%	39%

Using the 84 observations in the calibration set, the Research Model coefficient estimates are given below. The full analysis results are in Appendix D.

1. The Full RM estimated coefficients are:

$$PM = 2.51 \cdot KSLOC2^{1.03} \cdot PMAT^{1.42} \cdot PREC^{0.67} \cdot RESL^{-0.63} \cdot RELY^{1.06} \cdot DATA^{0.33} \cdot CPLX^{1.06} \cdot RUSE^{-0.74} \cdot DOCU^{0.22} \cdot RCON^{4.30} \cdot PERS^{2.63} \cdot AEXP^{-1.36} \cdot PEXP^{0.82} \cdot LTEX^{0.04} \cdot PCON^{0.58} \cdot TEAM^{0.90} \cdot FLEX^{-0.35} \cdot TOOL^{-0.77} \cdot SITE^{0.94} \cdot PVOL^{-0.21} \cdot SCED^{2.69}$$

2. The Reduced RM estimated coefficients are:

$$PM = 2.2 \cdot KSLOC2^{1.05} \cdot PMAT^{1.38} \cdot DATA^{0.41} \cdot CPLX^{1.4} \cdot RCON^{4.33} \cdot PERS^{1.9} \cdot TEAM^{1.03} \cdot SITE^{1.44} \cdot PVOL^{0.47} \cdot SCED^{2.56}$$

3. The Compact RM estimated coefficients are:

$$PM = 3.06 \cdot KSLOC2^{1.01} \cdot PMAT^{2.43} \cdot PROD^{0.7} \cdot DEVT^{0.39} \cdot ENVR^{0.63}$$

4. The Small RM estimated coefficients are:

$$PM = 3.06 \cdot KSLOC2^{1.01} \cdot PMAT^{2.46} \cdot EM^{0.61}$$

5. The Full COCOMO II estimated coefficients are:

$$PM = 2.33 \cdot SIZE^{0.95} \cdot SIZE^{(PREC \cdot 0.88)} \cdot SIZE^{(FLEX \cdot 0.44)} \cdot SIZE^{(RESL \cdot -1.69)} \cdot SIZE^{(TEAM \cdot 1.59)} \cdot SIZE^{(PMAT \cdot 3.64)} \cdot RELY^{0.53} \cdot DATA^{0.47} \cdot RUSE^{-0.55} \cdot DOCU^{-0.52} \cdot CPLX^{1.19} \cdot RCON^{2.92} \cdot PERS^{1.79} \cdot AEXP^{-0.61} \cdot PEXP^{0.91} \cdot LTEX^{-0.30} \cdot PCON^{-0.04} \cdot TOOL^{-0.31} \cdot SITE^{0.89} \cdot PVOL^{-0.13} \cdot SCED^{2.58}$$

6. The Reduced COCOMO II estimated coefficients are:

$$PM = 2.23 \cdot SIZE^{0.94} \cdot SIZE^{(TEAM \cdot 1.96)} \cdot SIZE^{(PMAT \cdot 3.23)} \cdot DATA^{0.38} \cdot CPLX^{1.25} \cdot RCON^{3.36} \cdot PERS^{1.48} \cdot SITE^{1.14} \cdot PVOL^{0.26} \cdot SCED^{2.55}$$

In all of the models PMAT was significant using the calibration data set.

5.9 Adding KPAs to the Research Model

There were two methods to assess PMAT on a project, see section 4.6.2 on page 46. Of the one hundred twelve observations, only fifty observations used KPA data to compute PMAT.

The distribution of the KPA data is interesting. Figure 18 below shows that for the SW-CMM Level 2 and 3 KPAs, the median is about the “Frequently” rating. This shows that good, successful companies have processes that already incorporate many of the Level 2 and 3 KPA goals. The Level 4 and 5 KPAs show a median of “About Half.” Histograms of each KPA are given in Appendix C.2.

A pairwise correlation analysis of the KPAs show two groups. KPA 2, Project Planning, and KPA 3, Project Tracking and Oversight are highly correlated, 0.82. This makes sense because the two areas support each other in planning and executing a project. For analysis purposes, KPAs 2 and 3 were combined into KPA 23 using their geometric mean.

KPA 7, Organizational Process Focus, and KPA 8, Organizational Process Definition, are highly correlated, 0.85. KPA 7 and KPA 10, Integrated Software Management, are correlated, 0.76. KPA 8 and KPA 10 are correlated, 0.72. The three KPAs are related. KPA 8 establishes a group to maintain process definitions, KPA 7, at the Organizational level.

KPA 10 uses the defined processes in managing software projects. For analysis purposes, KPAs 7, 8, and 10 were combined into KPA 7810 using their geometric mean.

	Almost Always	Frequently	About Half	Occasionally	Rarely if Ever	Don't Know or N/A	Total
KPA1: Requirements Management	26	14	7	3	0	62	112
KPA2: Software Project Planning	24	19	3	4	0	62	112
KPA3: Software Project Tracking and Oversight	24	13	9	4	0	62	112
KPA4: Software Subcontract Management	2	2	2	3	1	102	112
KPA5: Software Quality Assurance	19	17	9	4	0	63	112
KPA6: Software Configuration Management	22	23	5	0	0	62	112
KPA7: Organization Process Focus	20	10	7	5	8	62	112
KPA8: Organization Process Definition	21	13	7	3	6	62	112
KPA9: Training Program	10	14	10	10	3	65	112
KPA10: Integrated Software Management	18	13	7	5	7	62	112
KPA11: Software Product Engineering	15	27	3	4	1	62	112
KPA12: Intergroup Coordination	12	24	6	7	0	63	112
KPA13: Peer Reviews	15	15	4	9	4	65	112
KPA14: Quantitative Process Management	5	12	6	7	6	76	112
KPA15: Software Quality Management	6	9	8	9	6	74	112
KPA16: Defect Prevention	5	9	9	4	13	72	112
KPA17: Technology Change Management	6	7	5	5	15	74	112
KPA18: Process Change Management	8	10	3	8	11	72	112

Figure 18. KPA Distribution

The first thirteen KPAs were inserted into the Small Research Model with PMAT removed. In the Small RM, PMAT has an estimated exponent of 2.11 and a t-value of 4.77. KPA 4, Software Subcontract Management, was withheld because of the high number of "I Don't Know" responses. When inserting the KPAs directly into the Small Research Model all of the KPAs are insignificant, Figure 19. All of the estimation intervals include zero as an estimate.

There is not enough data to support analysis of individual KPA effects on effort. Using the statistician's rule of thumb, there are ninety possible responses with four observations required per response. That exceeds the current number of projects in the data set.

```
Data set = Db3_v11_970617_KPA
Response      = log[EFFORT]
Coefficient Estimates
Label         Estimate        Std. Error    t-value
Constant     -0.922021        1.26386        -0.730
log[KSLOC2]   1.06116         0.0983104      10.794
log[EM]       0.253494        0.169876        1.492
KPA1          0.427579        1.34149         0.319
KPA23        -1.50597         1.42907        -1.054
KPA5          0.105173        0.643145        0.164
KPA6         -0.167887        1.86325        -0.090
KPA7810       0.980771        1.07825         0.910
KPA9          0.112774        0.291503        0.387
KPA11         1.11771         1.10032         1.016
KPA12         0.998204        0.639690        1.560
KPA13        -0.516214        0.362163       -1.425

R Squared:               0.925114
Sigma hat:               0.519689
Number of cases:              50
Degrees of freedom:           38

Summary Analysis of Variance Table
Source          df      SS           MS          F     p-value
Regression      11   126.784      11.5259     42.68   0.0000
Residual        38   10.2629      0.270076
```

Figure 19. RM KPA Results

Chapter 6

CONCLUSIONS

6.1 Conclusions

1. For the one hundred twelve projects in this sample, Software Process Maturity was a significant factor affecting software development effort. After normalizing for the effects of other effort influences, a one-increment change in the rating of Process Maturity resulted in a 15% to 21% reduction in effort.
2. As an effort multiplier, PMAT's productivity range is between 2.04 and 2.59. This is not as high as PERS, Personnel Capability, but it is similar to CPLX, Software Product Complexity, and it is higher than the other predictor variable's effects.
3. The statistical significance of PMAT as an exponent predictor variable in the COCOMO II models was higher than for PMAT as a multiplicative predictor variable in the Research models. This suggests that it is appropriate to consider PMAT as an exponent predictor variable which acts to reduce diseconomies of scale, i.e., process maturity improvement savings are higher for large projects than on small projects.
4. Process Maturity should be in all cost models. The Capability Maturity Model is well defined. It establishes criteria to evaluate processes used to develop software. It provides a significant assessment of the effects of process on development effort.

6.2 Summary of Contributions

This research resulted in seven contributions to knowledge:

1. Demonstrated a method to distinguish the effects of an interesting factor from other factors affecting development effort.
2. Proposed a new specification for a Software Development Cost Model based on the econometric Log-Log model. The use of elasticities makes the model explainable and understandable.
3. The Compact Research Model demonstrates that aggregation of predictors eliminates their interaction while still producing a useful high-level model.
4. The steps used in this research for calibration, pruning, and independent variable insertion can be used to create an effort estimation model that is specific to a software development environment. This has the following advantages:

The data used to calibrate the model can be used to produce a prediction interval for the estimate.

The effects of local independent variables on effort are identified.

5. The calibration of the Research Model permits sensitivity analysis on the relationship of Process Maturity to other development effort factors. In other words, the relationship of the PMAT variable to other independent variables can be used to understand how the effects of Process Maturity can be offset by other factors.
6. The Research Model can be used to assist in calibration of the larger COCOMO II Post-Architecture model.
7. The data indicated that the Key Practices Areas of the Capability Maturity Model do enable a Software Organization to develop software with less effort.

6.3 Future Research

1. More KPA data needs to be gathered to assess which KPAs have the most influence on effort. Implementing the effort saving KPAs first would offset the costs of implementing the other KPAs. Based on the KPA results, the model could be refined to capture any nonuniform improvements in going from CMM Level n to Level (n+1).
2. This research established Process Maturity's effect on software development effort. This addresses software costs. A future investigation should study the effects Process Maturity has on software development schedule, i.e. cycletime.
3. The Log-Log model came from the field of Econometrics. It would be appealing to investigate the suitability of using other econometric models for use in the field of Software Engineering, e.g. Poisson model, Logit model.
4. The reports of CMM Return On Investment should be analyzed with the results of this research to determine if they confirm each other.
5. The negative coefficients for some of the predictors should be investigated using larger data samples and analysis of the effects of weak-dispersion. Not including these predictors leaves areas of influence uncovered.

ACRONYMS / GLOSSARY / SYMBOLS

Adj- R^2	R^2 adjusted for the number of parameters in the model.
ASLOC	Adapted Source Lines of Code
B_i	Population parameter coefficient.
b_i	Sample population parameter coefficient; an estimation of B_i.
BRAK	Code breakage; code developed but not used in the final product.
CMM	Capability Maturity Model
df	Degrees of freedom
DOD	Department of Defense
elasticity	The ratio of the change in the response variable to the change in the predictor variable.
k	Number of parameters or predictor variables in the model.
KPA	Key Process Area
KSLOC2	Thousands of Source Lines of Code. Includes new code and 20% of the adapted code.
IDA	Institute for Defense Analysis
n	Number of observations or projects.
NASA	National Aeronautics and Space Administration
NSLOC	New Source Lines of Code
p	Number of parameters in the model + 1
PE	Proportional Error (section 2.3.4 on page 17)
PM	Person Months (effort)
PRED(L)	The percentage of predictions that fall within L% of the actuals
R^2	Coefficient of determination. The amount of model explained variation in the data.
ROI	Return on Investment
SDC/CR	Software Development Capacity / Capacity Review
SEI	Software Engineering Institute
SPR	Software Productivity Research, Inc.
SSE	Sum of squares error.
SSR	Sum of squares regression
SST	Sum of squares total = SSE + SSR
SW-CMM	Software Capability Maturity Model
WBS	Work Breakdown Structure

REFERENCES

[Bate et al. 1994]. R. Bate, S. Garcia, J. Armitage, K. Cusick, R. Jones, D. Kuhn, I. Minnich, H. Pierson, T. Powell, and A. Reichner, A Systems Engineering Capability Maturity Model, Version 1.0," CMU/SEI-94-HB-04, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1994.

[Boehm 1981]. B.W. Boehm, Software Engineering Economics, Prentice-Hall, NJ, 1981.

[Boehm 1987]. B.W. Boehm, "Improving Software Productivity," IEEE Computer, August 1987, pp. 34-57.

[Boehm 1993]. B.W. Boehm, "Economic Analysis of Software Technology Investments," Analytical Methods in Software Engineering Economics, T.R. Gulledge and W.P. Hutzler (Eds.), Springer-Verlag, New York, NY, 1993, pp.1-37.

[Boehm et al. 1995]. B.W. Boehm, B Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost models for future software life cycle processes: COCOMO 2.0," Annals of Software Engineering, J.D. Arthur and S.M. Henry (Eds.), J.C. Baltzer AG, Science Publishers, Amsterdam, The Netherlands, 1995, pp. 57-94.

[Brodman and Johnson 1995]. J.G. Brodman and D.L. Johnson, "Return on Investment (ROI) from Software Process Improvement as Measured by US Industry," Software Process Improvement and Practice, John Wiley &Sons Ltd., Sussex, England and Gauthier-Villars, 1995, pp. 35-47.

[Butler 1995]. K. Butler, "The Economic Benefits of Software Process Improvement," Crosstalk, Hill AFB, Ogden, Ut., 1995, pp. 14-17.

[Crosby 1984]. P. Crosby, Quality Without Tears, McGraw-Hill, New York, NY, 1986.

[Conte et al. 1986]. S. Conte, H. Dunsmore, and V. Shen, Software Engineering Metrics and Models, Benjamin/Cummings, Menlo Park, Ca., 1986.

[Curtis et al. 1995]. B. Curtis, W. Hefley, and S. Miller, "People Capability Maturity Model," CMU/SEI-95-MM-02, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., September 1995.

[Dion 1993]. R. Dion, "Process Improvement and the Corporate Balance Sheet," IEEE Software, October 1993, pp. 28-35.

[Ferguson et al. 1996]. J. Ferguson, J. Cooper, M. Falat, M. Fisher, A. Guido, J. Marciniak, H. Matejceck, R. Webster, "Software Acquisition Capability Maturity Model (SA-CMM), Version 1.01," CMU/SEI-96-TR-020, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1996.

[Griffiths et al. 1993]. W.E. Griffiths, R.C. Hill, and G.G. Judge, Learning and Practicing Econometrics. John Wiley & Sons, Inc., New York, NY, 1993.

[Gulezian 1986]. R. Gulezian, "Utilizing COCOMO Inputs as a Basis for Developing Generalized Software Development Cost Estimation Models," COCOMO / WICOMO User's Group Meeting, Wang Institute, Tyngsboro, Ma., May 1986.

[Hair et al. 1995]. J. Hair, R. Anderson, R. Tatham, and W. Black, Multivariate Data Analysis with Readings, Prentice Hall, Englewood Cliffs, N.J., 1995.

[Herbsleb et al. 1994]. J. Herbsleb, A. Carleton, J. Rozum, J. Siegel, D. Zubrow, "Benefits of CMM-Based Software Process Improvement: Initial Results," CMU/SEI-94-TR-13, Software Engineering Institute, Pittsburgh, Pa., 1994.

[Herbsleb et al. 1997]. J. Herbsleb, D. Zubrow, D. Goldenson, W. Hayes, and M. Paulk, "Software Quality and the Capability Maturity Model," Communications of the ACM, June 1997, pp. 30-40.

[Humphrey and Sweet 1987]. W.S. Humphrey and W.L. Sweet, "A Method for Assessing the Software Engineering Capability of Contractors," CMU/SEI-87-TR-23, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1987.

[Humphrey 1989]. W.S. Humphrey, Managing the Software Process, Addison-Wesley, Reading, Ma. 1989.

[Humphrey et al. 1991]. W.S. Humphrey, T.R. Snyder, and RR. Willis, "Software Process Improvement at Hughes Aircraft," IEEE Software, August 1991, pp. 11-23.

[Kemerer 1987]. C. Kemerer, "An Empirical Validation of Software Cost Estimation Models," Communications of the ACM, May 1987, pp.416-429.

[Kitchenham 1990]. B. Kitchenham, "Software Development Cost Models," in Software Reliability Handbook, R. Rook (Ed.), Elsevier, London, U.K., 1990.

[Krishnan 1996]. M. Krishnan, Cost and Quality Considerations in Software Product Management, Dissertation, Graduate School of Industrial Administration, Carnegie Mellon University, 1996.

[Madachy 1996]. R. Madachy, "Systems Dynamics Modeling of an Inspection-Based Process," Proceedings of the 18th International Conference on Software Engineering, Berlin, Germany, March 1996, pp.376-386.

[McGibbon 1996]. T. McGibbon, "A Business Case for Software Process Improvement," Contract Number F30602-92-C-0158, Data & Analysis Center for Software (DACS), Kaman Sciences Corp., Utica, NY, 1996.

[Paulk et al. 1993] M. Paulk, B. Curtis, M. Chrissis, and C. Weber, "Capability Maturity Model for Software, Version 1.1," CMU/SEI-93-TR-24, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1993.

[Paulk et al. 1995a]. M.C. Paulk, C.V. Weber, B. Curtis, and M.B. Chrissis. The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley, Reading, Ma., 1995.

[Paulk 1995b]. M. Paulk, "How ISO 9001 Compares with the CMM," IEEE Software, January 1995, pp74-83.

[Paulk 1997]. M. Paulk, "Software Capability Maturity Model, Version 2," Draft Technical Report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., January 1997.

[Peterson 1997]. W.C. Peterson, "SEI's Software Process Program," Presentation to the Board of Visitors, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., April 1997.

[PRICE S 1993]. PRICE S Reference Manual, PRICE Systems, Moorestown, N.J., January 1993.

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

[Reifer et al. 1989]. D. Reifer, P. Kane, and D. Willens, SoftCost-R User Guide, Reifer Consultants, Inc., Torrance, Ca. 1989.

[Rubin 1983]. H. Rubin, "Macroestimation of Software Development Parameters: the Estimacs System," in SOFTFAIR Conference on Software Development Tools, Techniques and Alternatives, Arlington, IEEE Press, New York, NY, July 1983, pp.4-16.

[Saiedian and Kuzara 1995]. H. Saiedian and R. Kuzara, "SEI Capability Maturity Model's Impact on Contractors," IEEE Computer, January 1995, pp.16-26.

[SEER-SEM 1994]. SEER-SEM User's Manual, Galorath Associates, Inc., Los Angeles, Ca., 1994.

[SEL 1990]. Manager's Handbook for Software Development, SEL-84-101, National Aeronautics and Space Administration Goddard Space Flight Center, Greenbelt, Md, 1990.

[SLIM 1995] SLIM 3.2 User's Manual, Quantitative Software Measurement, Inc., McLean, Va., 1995.

[SPR 1994]. "Checkpoint Assessment Questionnaire," Software Productivity Research, Inc., Burlington, Ma., 1994.

[Springsteen et al. 1992]. B. Springsteen, B. Brykczynski, D. Fife, R. Meeson, and J. Norris, "Policy Assessment for the Software Process Maturity Model," IDA D-1202, Institute for Defense Analyses, Alexandria, Va., 1992.

[Tausworthe 1981]. R.C. Tausworthe, Deep Space Network Software Cost Estimation Model, JPL Publication 81-7, Jet Propulsion Laboratory, Pasadena, Ca., April 1981.

[Weisberg 1985]. S. Weisberg, Applied Linear Regression, John Wiley & Sons, New York, NY, 1985.

[Wohlwend and Rosenbaum 1994]. H. Wohlwend and S. Rosenbaum, “Schlumberger’s Software Improvement Program,” IEEE Transactions on Software Engineering, November 1994, pp. 833-839.

Appendix A

RATIONALE FOR A PROCESS'S MATURITY INFLUENCE ON EFFORT

Table 19 shows a summary of the effect each KPA has on effort and the reduction of rework. The symbols (+,-) used in the table are from the perspective of effort. A **plus (+)** shows an effect that requires extra effort and a **minus (-)** shows an effect that reduces rework. Iteration of a symbol (e.g., ++) indicates a stronger effect. All the KPAs are defined with their goals and practices in [Paulk et al. 1995].

Analysis of the effect each KPA has on effort leads to the following observations:

- Most KPA's will probably produce a net savings in effort as seen by the *reduction of rework* analysis for each KPA. In the Plans and Requirements stage, almost all KPA's require more effort. In later stages they reduce effort by eliminating the ripple effect of errors, ensuring the necessary tools and plans are available when needed, and coordinating later life-cycle activities for maximum effectiveness.
- Some KPA's are likely to produce considerably more savings than others. For instance Peer Reviews is an up-front method for eliminating errors early in the life-cycle. The earlier the elimination of errors the greater the savings in extra effort. Madachy's analysis of several hundred peer reviews on a Litton project indicated that their use reduced effort by about 10% [Madachy 1996]. Most effort savings from KPA's will come from *reducing rework* [Dion 1993]. KPA's such as Training, Product Engineering, Quantitative Process Measurement and Technology Change Management reduce effort by increasing the skill of the work force, ensuring software development of the product is achievable, spreading *best practices* across the organization, and incorporating change the will improve product quality.
- Rework savings from several KPA's overlap considerably. For instance if Requirements Management is implemented before Configuration Management then extra effort will be required to baseline the requirements. Whereas if these KPA implementations are reversed, the procedures for baselining the requirements will already be in place. This will complicate Blackbox analysis of individual KPA effort reduction contributions.
- While not mentioned in this analysis, case studies cited earlier [Dion 1993, Herbsleb et. al. 1994] have reported an increase in morale. This will reduce effort because people assimilate the practices required by in the KPA's and without enforcement, they enact them.
- NOTE: There is a saying that goes "I would have written you a shorter letter but

I did not have the time” which reflects a paradox that more effort could result in a smaller software product (the accepted premise is that the larger the product the more effort required to produce it if writing all new code). It is easier to write big, sloppy code than tight, concise code. The presence of this anomaly will be suggested by the use of software development process such as the spiral model or iterative model where early effort can be used to refined and verify later life-cycle requirements.

Table 19: KPA vs. Development Stage

KPA	Plans and Requirements	Design	Code and Unit Test	Integration and Test
KPA 1: Requirements Management	+ Extra effort to establish requirements baseline and set up management process.	+ Extra effort to review modifications to requirements before incorporation into product. -- Reduction of ripple effect of changes to poorly-assessed requirements. - Reduction in code size due to carefully defined requirements.		
KPA2: Software Project Planning	+ Establish plans, prepare for down-stream activities.	+ Creation of test plans. - Availability of tools to support design.	-- Availability of tools and facilities to support code, unit test and integration and test.	
		-- Rework reduced because development activities are coordinated within constraints.		
KPA 3: Software Project Tracking and Oversight	+ Extra effort to establish tracking and oversight functions	+ Extra effort to perform tracking, communication of status and revising plans -- Rework reduced due to early detection of actions that need correction.		
KPA 4: Software Subcontract Management	+ Extra work to select subcontractor and set up management process.	+ Extra work in tracking, reviewing, and changing commitments with subcontractor's performance and results. -- Rework reduced due to early detection and correction of contractor-subcontractor incompatibilities.		
KPA 5: Software Quality Assurance	+ Extra work to establish standards and procedures	+ Extra effort to review project activities and audit software work products for conformance. -- Rework reduced due to early identification of non-conformances.		
KPA 6: Software Configuration Management	+ Extra work in creating baselines and setting up procedures.	+ Extra work in maintaining requirements, design, code and test baselines. -- Rework reduced due to avoidance of uncoordinated changes, incompatible or lost baselines.		
KPA 7: Organizational Process Focus	+ Extra effort to establish organizational responsibility for software process activities that improve the overall software process capability. + Extra effort to assess, develop, maintain, and impact organization's and project's software process. - Rework reduced due to reduced process inconsistencies across projects.			

Table 19: KPA vs. Development Stage

KPA	Plans and Requirements	Design	Code and Unit Test	Integration and Test
KPA 8: Organizational Process Definition	+ Extra effort in collecting, documenting, and maintaining requirements analysis activities.	+ Extra effort in collecting, documenting and maintaining design activities, test plan activities, QA activities, and CM activities.	+ Extra effort in collecting, documenting, and maintaining coding standards and unit test procedures.	+ Extra effort in collecting, documenting, and maintaining integration procedures and procedures for execution of test plans.
	+ Extra effort in monitoring, evaluating, and disseminating of new processes, methods, and tools.			
KPA 9: Training Program	+ Extra effort in identifying training needed by organization, project or individuals for each activity. + Extra effort required for on-the-job training during different activities. - Rework reduced by having people trained in quality assurance and configuration management activities. -- General effort reductions due to training in applications, tools, techniques, languages.			
	- Rework reduced by having skilled people perform requirements analysis.	- Rework reduced by having skilled people perform design and planning of tests.	- Rework reduced by having skilled people perform coding and unit testing.	- Rework reduced by having skilled people perform integration and testing.
KPA 10: Integrated Software Management	- Rework reduced by adopting standard processes for requirements analysis, configuration management, quality assurance, test planning, and risk management.	- Rework reduced by adoption of a standard design activity.	-Rework reduced by adoption of coding standards and unit test procedures.	- Rework reduced by adoption of predefined integration and test activities.
	+ Extra effort in evaluating and codifying best practices. - Rework reduced by adopting best practices in staffing, planning, tracking, and estimating a software project. - Rework reduced by anticipating problems and acting to prevent or minimize the effects of the problems.			
KPA 11: Software Product Engineering	+ Extra effort in evaluating and codifying best practices. + Extra effort expended on careful design and code reduces product size. - Rework is reduced because consistency is maintained across software products permitting experiences in one product to be used in another. -- General effort reductions due to improved tools and techniques.			
KPA 12: Intergroup Coordination	+ Extra effort required to plan and manage interfaces and interactions between groups.			
	- rework reduced because realistic software requirements are set.	- Rework reduced because design assumptions about the target platform can be confirmed.	- Rework reduced because code dependencies on hardware interfaces can be confirmed.	- Rework reduced due to earlier detection and correction of software/system incompatibilities.

Table 19: KPA vs. Development Stage

KPA	Plans and Requirements	Design	Code and Unit Test	Integration and Test
KPA 13: Peer Review	+ Extra effort required for planning and managing review process.			
	- Rework reduced by early removal of requirements defects.	-- Rework reduced by early removal of requirements and design defects.	-- Rework reduced by early removal of requirements, design, and code defects.	-- Rework reduced by early removal of requirements, design, and code defects.
KPA 14: Quantitative Process Management	+ Extra effort to define goals, collect and analyze performance data, and make adjustments to maintain process performance. + Extra effort to disseminate the results and baseline the performance of the process. -- Rework reduced due to process improvements from analysis of previous projects.			
KPA 15: Software Quality Management	+ Extra effort required to define quality goals. + Extra effort required to establish, monitor and adjust plans, products, and activities. -- Rework reduced because system, component, or process meets customer or end-user needs or expectations, or due to quality improvements from analysis of previous projects.			
KPA 16: Defect Prevention	+ Extra effort in identifying defects, performing causal analysis to determine the root cause and assess implications of the defects for future activities. Extra effort in disseminating status and lessons learned. -- Rework reduced by eliminating causes that produce defects. Earlier elimination results in greater reduction of rework.			
KPA 17: Technology Change Management	+ Extra effort in identifying, selecting, and evaluating new technologies, and incorporating effective technologies into the organization. -- Rework reduced where technology reduces defect detection and correction effort, improves coordination, reduces product complexity, or stabilizes product platform. -- General effort reductions due to improved tools and techniques.			
KPA 18: Process Change Management	+ Extra effort required in identifying, evaluating, and implementing improvements to the organization standard software process. -- Rework reduced due to more rapid incorporation of new practices that among other things prevent rework.			

The different activities used in Table 19 are described below [Boehm 1981]:

- Plans and Requirements. During this stage system requirements are allocated to hardware and software. The concept of operation for the system is specified (human - machine interactions are understood and defined). The software requirements are validated for completeness, consistency, testability, and feasibility for functional, performance and interface specifications.

Life-cycle plans are created to address:

  + Project milestones and detailed schedules
  + Project resources and budgets
  + Customer, developer, and end-user responsibilities; project organizational structure
  + Product control such as configuration management, quality assurance, risk management, development standards, and a software development model (waterfall, iterative, spiral) which addresses "when" and "for how long" to perform design, coding, and testing.
  + Training in activities and techniques; software development support products

- Design. This stage produces a product design specification broken down from the system level to the sub-system then component then unit levels where a unit is a well defined piece of the product about 100 - 300 lines of source code in size. If appropriate, the different software builds are identified. This specification includes control structure, data flow and component interfaces. The design is verified for completeness and consistency and validated to the requirements. High-risk development issues are identified and resolved. Two test plans, integration (for verification of the software) and acceptance (for validation to the requirements) are drafted and approved.
- Code and Unit Test. This stage produces coding for all units identified for the software build. The units are verified for nominal and extreme inputs, error handling, all data declarations / initialization, executable statements and all branching options. The code is verified for compliance to programming standards. Documentation for the units is completed.
- Integration and Test. This stage involves integration of units into components and components into sub-systems. Components and sub-systems are tested on the target platform for correct outputs, error handling, and desired performance using nominal and extreme inputs and different platform configurations. It also includes integration, test, and acceptance of the software system and deliverables (manuals, reports, code).

Appendix B

COCOMO II COST ESTIMATION QUESTIONNAIRE

B.1 Introduction

The Center for Software Engineering at the University of Southern California is conducting research to update the software development cost estimation model called COCOMO [Boehm 1981]. The project name is COCOMO II and is led by Dr. Barry W. Boehm [Boehm et al. 1995].

A fundamental requirement for such research is real-world software development project data. This data will be used to test hypotheses and verify the model's postulations. In return the model will be open and made available to the public. The contribution of your data will ensure the final model is useful.

The data that is contributed is important to us. We will safeguard your contribution so as not to compromise company proprietary information. The next section discusses the data management aspects of the project. The following section is the data collection form. The last section is an explanation of expected values in the data collection form.

Some Affiliates have an active collection program and the data from past projects is available for the COCOMO II data collection effort. This questionnaire can be used to extract relevant COCOMO II data.

This questionnaire attempts to address two different levels of data granularity: project level and component level. The project level of granularity is data that is applicable for the whole project. This includes things like application type and development activity being reported. Component level data are things like size, cost, and component cost drivers. If the data being submitted is on a project that has multiple components then fill out the project data once, and the component data for each of the identifiable component. If the data are being submitted for the whole project fill out the form once.

B.2 Project Level Information

General Information

B.2.1 Affiliate Identification Number. Each separate software project contributing data will have a separate file identification number of the form XXX. XXX will be one of a random set of three-digit organization identification numbers, provided by USC Center for Software Engineering to the Affiliate.

_____

B.2.2 Project Identification Number. The project identification is a three digit number assigned by the organization. Only the Affiliate knows the correspondence between *YYY* and the actual project. The same project identification must be used with each data submission.

_____

B.2.3 Application Type. This field captures a broad description of the type of activity this software application is attempting to perform.

Circle One:

Command and Control	MIS	Simulation
Communication	Operating Systems	Software Dev. Tools
Diagnostics	Process Control	Testing
Engineering and Science	Signal processing	Utilities

Other: _____

B.2.4 Activity. This field captures the phase of development that the project is in. For one-time reporting the activity is 'completed'. It is assumed that data for completed projects includes data from software requirements through integration / test. Please report the correct phasing if this is not the case.

Circle One:	Requirements	Design	Code
	Unit Test	Integration/Test	Maintenance
	Completed		

Other: _____

B.2.5 Development Process. This is a description of the software process used to control the software development.

_____

B.2.6 Development Process Iteration. If the process is iterative, e.g. spiral, which iteration is this?

_____

B.2.7 COCOMO Model. This specifies which COCOMO II model is being used in this data submission. If this is a "historical" data submission, select the Post-Architecture model or the Applications Composition model.

- Application Composition: This model involves prototyping efforts to resolve potential high risk issues such as user interfaces, software/system interaction, performance, or technology maturity.
- Early Design: This model involves exploration of alternative software/system architectures and concepts of operations. At this stage of development, not enough is known to support fine-grain cost estimation.
- Post-Architecture: This model involves the actual development and maintenance of a software product. This stage of development proceeds most cost-effectively if a software life-cycle architecture has been developed; validated with respect to the system's mission, concept of operation, and risk; and established as the framework for the product.

Circle One:Application Composition, Early Design, Post-Architecture

Schedule

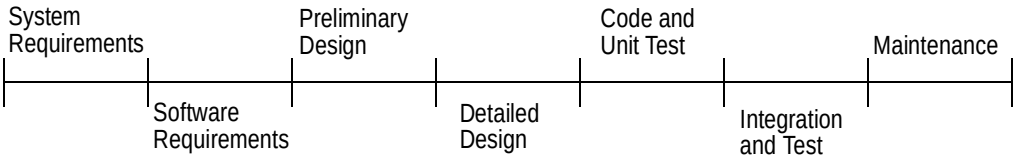
B.2.8 Year of development. For reporting of historical data, please provide the year in which the software development was completed. For periodic reporting put the year of this submission or leave blank.

_____

B.2.9 Schedule Months. For reporting of historical data, provide the number of calendar months from the time the development began through the time it completed (from the beginning of Software Preliminary Design through the end of System Test and Integration). For periodic reporting, provide the number of months in this development activity.

_____

Circle the life-cycle phases that the schedule covers:



- Software Requirements. This phase defines the complete, validated specification of the required functions, interfaces, and performances for the software product.
- Preliminary Design. This phase defines the complete, verified specification of the overall hardware/software architecture, control structure, and data structure for the product, along with such necessary components as draft user's manuals and test plans.
- Detailed Design. This phase defines a complete, verified specification of the control structure, data structure, interface relations, sizing, key algorithms, and assumptions of each program component.
- Code & Unit Test. This phase produces a complete and verified set of program components.
- S/W System Integration & Test. This phase produces a properly functioning software product composed of the software components.
- Maintenance. This phase produces a fully functioning update of the hardware/ software system.

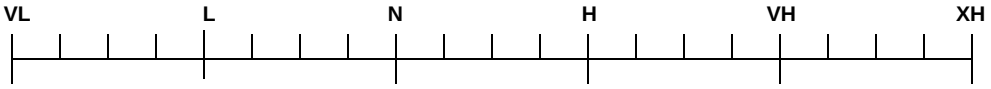
Project Exponential Cost Drivers

Scale Factors (W_i)	Very Low	Low	Nominal	High	Very High	Extra High
Precedented-ness	thoroughly unprece-dented	largely unprece-dented	somewhat unprece-dented	generally familiar	largely familiar	throughly familiar
Development Flexibility	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
Architecture / risk resolution[a]	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
Team cohesion	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions

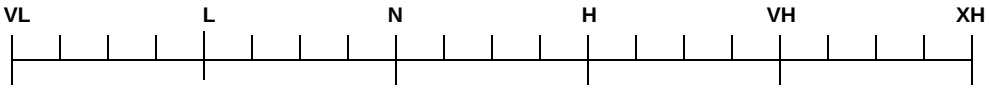
[a.] % significant module interfaces specified,% significant risks eliminated.

Enter the rating level for the first four cost drivers by circling one of the tick marks.

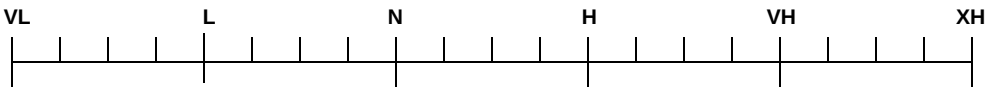
B.2.10 Precedentedness (PREC). If the product is similar to several that have been developed before then the precedentedness is high.



B.2.11 Development Flexibility (FLEX). This cost driver captures the amount of constraints the product has to meet. The more flexible the requirements, schedules, interfaces, etc., the higher the rating. See the User's Manual for more details.

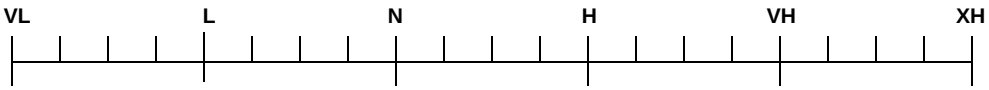


B.2.12 Architecture / Risk Resolution (RESL). This cost driver captures the thoroughness of definition and freedom from risk of the software architecture used for the product. See the User's Manual for more details.



B.2.13 Team Cohesion (TEAM). The Team Cohesion cost driver accounts for the sources of project turbulence and extra effort due to difficulties in synchronizing the

project's stakeholders: users, customers, developers, maintainers, interfacers, others. See the User's Manual for more details.



B.2.14 Process Maturity (PMAT). The procedure for determining PMAT is organized around the Software Engineering Institute's Capability Maturity Model (CMM). The time period for reporting process maturity is at the time the project was underway. We are interested in the capabilities practiced *at the project level* more than the overall organization's capabilities.

There are three ways of responding to this question: choose only one. "Key Process Area Evaluation" requires a response for each Key Process Area (KPA). We have provided enough information for you to self-evaluate the project's enactment of a KPA (we hope will you will take the time to complete this section). "Overall Maturity Level" is a response that captures the result of an organized evaluation based on the CMM. "No Response" means you do not know or will not report the process maturity either at the Capability Maturity Model or Key Process Area level.

- ☐ **No Response**

Overall Maturity Level

- ☐ CMM Level 1 (lower half)
- ☐ CMM Level 1 (upper half)
- ☐ CMM Level 2
- ☐ CMM Level 3
- ☐ CMM Level 4
- ☐ CMM Level 5

Basis of estimate:

- ☐ Software Process Assessment (SPA)
- ☐ Software Capability Evaluation (SCE)
- ☐ Interim Process Assessment (IPA)
- ☐ Other: _____

Key Process Area Evaluation

Enough information is provided in the following table so that you can assess the degree to which a KPA was exercised on the project. Each KPA is briefly described and its goals are given. The response catagories are explained below:

- Almost Always (over **90%** of the time) when the goals are consistently achieved and are well established in standard operating procedures.
- Frequently (about **60 to 90%** of the time) when the goals are achieved relatively often, but sometimes are omitted under difficult circumstances.
- About Half (about **40 to 60%** of the time) when the goals are achieved about half of the time.
- Occasionally (about **10 to 40%** of the time) when the goals are sometimes achieved, but less often.
- Rarely If Ever (less than **10%** of the time) when the goals are rarely if ever achieved.
- Does Not Apply when you have the required knowledge about your project or organization and the KPA, but you feel the KPA does not apply to your circumstances (e.g. Subcontract Management).
- Don't Know when you are uncertain about how to respond for the KPA.

Key Process Area	Goals of each KPA	Almost Always	Frequently	About Half	Occasionally	Rarely If Ever	Does Not Apply	Don't Know
Requirements Management: involves establishing and maintaining an agreement with the customer on the requirements for the software project.	System requirements allocated to software are controlled to establish a baseline for software engineering and management use. Software plans, products, and activities are kept consistent with the system requirements allocated to software.	☐	☐	☐	☐	☐	☐	☐
Software Project Planning: establishes reasonable plans for performing the software engineering activities and for managing the software project.	Software estimates are documented for use in planning and tracking the software project. Software project activities and commitments are planned and documented. Affected groups and individuals agree to their commitments related to the software project.	☐	☐	☐	☐	☐	☐	☐
Software Project Tracking and Oversight: provides adequate visibility into actual progress so that management can take corrective actions when the software project's performance deviates significantly from the software plans.	Actual results and performances are tracked against the software plans. Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans. Changes to software commitments are agreed to by the affected groups and individuals.	☐	☐	☐	☐	☐	☐	☐
Software Subcontract Management: involves selecting a software subcontractor, establishing commitments with the subcontractor, and tracking and reviewing the subcontractor's performance and results.	The prime contractor selects qualified software subcontractors. The prime contractor and the software subcontractor agree to their commitments to each other. The prime contractor and the software subcontractor maintain ongoing communications. The prime contractor tracks the software subcontractor's actual results and performance against its commitments.	☐	☐	☐	☐	☐	☐	☐

Key Process Area	Goals of each KPA	Almost Always	Frequently	About Half	Occasionally	Rarely If Ever	Does Not Apply	Don't Know
Software Quality Assurance: provides management with appropriate visibility into the process being used by the software project and of the products being built.	Software quality assurance activities are planned. Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively. Affected groups and individuals are informed of software quality assurance activities and results. Noncompliance issues that cannot be resolved within the software project are addressed by senior management.	☐	☐	☐	☐	☐	☐	☐
Software Configuration Management: establishes and maintains the integrity of the products of the software project throughout the project's software life cycle.	Software configuration management activities are planned. Selected software work products are identified, controlled, and available. Changes to identified software work products are controlled. Affected groups and individuals are informed of the status and content of software baselines.	☐	☐	☐	☐	☐	☐	☐
Organization Process Focus: establishes the organizational responsibility for software process activities that improve the organization's overall software process capability.	Software process development and improvement activities are coordinated across the organization. The strengths and weaknesses of the software processes used are identified relative to a process standard. Organization-level process development and improvement activities are planned.	☐	☐	☐	☐	☐	☐	☐
Organization Process Definition: develops and maintains a usable set of software process assets that improve process performance across the projects and provides a basis for cumulative, long- term benefits to the organization.	A standard software process for the organization is developed and maintained. Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.	☐	☐	☐	☐	☐	☐	☐

Key Process Area	Goals of each KPA	Almost Always	Frequently	About Half	Occasionally	Rarely If Ever	Does Not Apply	Don't Know
Training Program: develops the skills and knowledge of individuals so they can perform their roles effectively and efficiently.	Training activities are planned. Training for developing the skills and knowledge needed to perform software management and technical roles is provided. Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles.	☐	☐	☐	☐	☐	☐	☐
Integrated Software Management: integrates the software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets.	The project's defined software process is a tailored version of the organization's standard software process. The project is planned and managed according to the project's defined software process.	☐	☐	☐	☐	☐	☐	☐
Software Product Engineering: integrates all the software engineering activities to produce and support correct, consistent software products effectively and efficiently	The software engineering tasks are defined, integrated, and consistently performed to produce the software. Software work products are kept consistent with each other.	☐	☐	☐	☐	☐	☐	☐
Intergroup Coordination: establishes a means for the software engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs effectively and efficiently.	The customer's requirements are agreed to by all affected groups. The commitments between the engineering groups are agreed to by the affected groups. The engineering groups identify, track, and resolve intergroup issues.	☐	☐	☐	☐	☐	☐	☐
Peer Review: removes defects from the software work products early and efficiently.	Peer review activities are planned. Defects in the software work products are identified and removed.	☐	☐	☐	☐	☐	☐	☐

Key Process Area	Goals of each KPA	Almost Always	Frequently	About Half	Occasionally	Rarely If Ever	Does Not Apply	Don't Know
Quantitative Process Management: controls the process performance of the software project quantitatively.	The quantitative process management activities are planned. The process performance of the project's defined software process is controlled quantitatively. The process capability of the organization's standard software process is known in quantitative terms.	☐	☐	☐	☐	☐	☐	☐
Software Quality Management: involves defining quality goals for the software products, establishing plans to achieve these goals, and monitoring and adjusting the software plans, software work products, activities, and quality goals to satisfy the needs and desires of the customer and end user.	The project's software quality management activities are planned. Measurable goals for software product quality and their priorities are defined. Actual progress toward achieving the quality goals for the software products is quantified and managed.	☐	☐	☐	☐	☐	☐	☐
Defect Prevention: analyzes defects that were encountered in the past and takes specific actions to prevent the occurrence of those types of defects in the future.	Defect prevention activities are planned. Common causes of defects are sought out and identified. Common causes of defects are prioritized and systematically eliminated.	☐	☐	☐	☐	☐	☐	☐
Technology Change Management: involves identifying, selecting, and evaluating new technologies, and incorporating effective technologies into the organization.	Incorporation of technology changes are planned. New technologies are evaluated to determine their effect on quality and productivity. Appropriate new technologies are transferred into normal practice across the organization.	☐	☐	☐	☐	☐	☐	☐
Process Change Management: involves defining process improvement goals and, with senior management sponsorship, proactively and systematically identifying, evaluating, and implementing improvements to the organization's standard software process and the projects' defined software processes on a continuous basis.	Continuous process improvement is planned. Participation in the organization's software process improvement activities is organization wide. The organization's standard software process and the projects' defined software processes are improved continuously.	☐	☐	☐	☐	☐	☐	☐

B.3 Component Level Information

Component ID

If the whole project is being reported as a single component then skip to the next section.

If the data being submitted is for multiple components that comprise a single project then it is necessary to identify each component with its project. Please fill out this section for each component and attach all of the component sections to the project sections describing the overall project data.

B.3.1 Affiliate Identification Number. Each separate software project contributing data will have a separate file identification number of the form XXX. XXX will be one of a random set of three-digit organization identification numbers, provided by USC Center for Software Engineering to the Affiliate.

_____

B.3.2 Project Identification Number. The project identification is a three digit number assigned by the organization. Only the Affiliate knows the correspondence between *YYY* and the actual project. The same project identification must be used with each data submission.

_____

B.3.3 Component Identification (if applicable). This is a unique sequential **letter** that identifies a software module that is part of a project.

Circle One: A B C D E F G H I

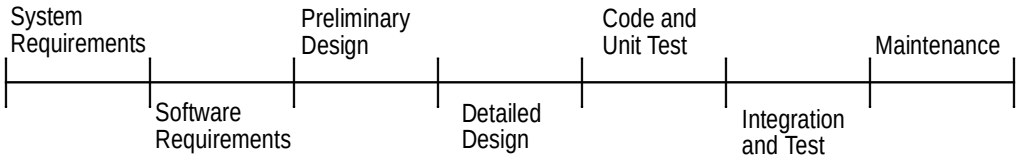
J K L M N O P Q R

Cost

B.3.4 Total Effort (Person Months). For one-time reporting, provide the effort in Person Months associated with development and test of the software component described, including its share of such common activities as system design and integration. For periodic reporting, provide the effort in Person Months since the project began.

_____

Circle the life-cycle phases that the effort estimate covers:



B.3.5 Hours / Person Month. Indicate the average number of hours per person month experienced by your organization.

_____

B.3.6 Labor Breakout. Indicate the percentage of labor for different categories,e.g. Managers, S/W Requirement Analysts, Designers, CM/QA Personnel, Programmers, Testers, and Interfacers for each phase of software development:

Labor Category	**Total for all phases**	Rqts	PD	DD	CUT	IT	M

- Requirements(Rqts). This phase defines the complete, validated specification of the required functions, interfaces, and performances for the software product.
- Preliminary Design (PD). This phase defines the complete, verified specification of the overall hardware/software architecture, control structure, and data structure for the product, along with such necessary components as draft user's manuals and test plans.
- Detailed Design (DD). This phase defines a complete, verified specification of the control structure, data structure, interface relations, sizing, key algorithms, and assumptions of each program component.

- Code & Unit Test (CUT). This phase produces a complete and verified set of program components.
- S/W System Integration & Test (IT). This phase produces a properly functioning software product composed of the software components.
- Maintenance (M). This phase produces a fully functioning update of the hardware/software system.

Size

The project would like to collect size in object points, logical lines of code, and unadjusted function points. Please submit all size measures that are available, e.g. if you have a component in lines of code and unadjusted function points then submit both numbers.

B.3.7 Percentage of Code Breakage. This is an estimate of how much the requirements have changed over the lifetime of the project. It is the percentage of code thrown away due to requirements volatility. For example, a project which delivers 100,000 instructions but discards the equivalent of an additional 20,000 instructions would have a breakage of value of 20. See the User's Manual for more detail.

_____

B.3.8 Object Points. If the COCOMO II Applications Programming model was used then enter the object point count.

_____

B.3.9 New Unique SLOC. This is the number of new source lines of code (SLOC) generated.

_____

B.3.10 SLOC Count Type. When reporting size in source lines of code, please indicate if the count was for *logical* SLOC or *physical* SLOC. If both are available, please submit both types of counts. If neither type of count applies to the way the code was counted, please describe the method. An extensive definition for logical source lines of code is given in an Appendix in the Model User's Manual.

Circle One:

Logical SLOC	Physical SLOC (carriage returns)
Physical SLOC (semicolons)	Non-Commented/Non-Blank SLOC

Other: _____

B.3.11 Unadjusted Function Points. If you are using the Early Design or Post-Architecture model, provide the total Unadjusted Function Points for each type. An Unadjusted Function Point is the product of the function point count and the weight for that type of point. Function Points are discussed in the User's Manual.

_____

B.3.12 Programming Language. If you are using the Early Design or Post-Architecture model, enter the language name that was used in this component, e.g. Ada, C, C++, COBOL, FORTRAN and the amount of usage if more than one language was used.

Language Used	Percentage Used

B.3.13 Software Maintenance Parameters. For software maintenance, use items 4.8 - 4.12 to describe the size of the base software product, and use the same units to describe the following parameters:

a. Amount of software added: _____

b. Amount of software modified: _____

c. Amount of software deleted: _____

B.3.14 Object Points Reused. If you are using the Application Composition model, enter the number of object points reused. Do not fill in the fields on DM, CM, IM, SU, or AA.

_____

B.3.15 ASLOC Adapted. If you are using the Early Design or Post-Architecture model enter the amounts for the SLOC adapted.

_____

B.3.16 ASLOC Count Type. When reporting size in source lines of code, please indicate if the count was for *logical* ASLOC or *physical* ASLOC. If both are available,

please submit both types of counts. If neither type of count applies to the way the code was counted, please describe the method. An extensive definition for logical source lines of code is given in an Appendix in the Model User's Manual.

Circle One:

Logical ASLOC	Physical ASLOC (carriage returns)
Physical ASLOC (semicolons)	Non-Commented/Non-Blank ASLOC

Other: _____

B.3.17 Design Modified - DM. The percentage of design modified.

_____

B.3.18 Code Modified - CM. The percentage of code modified.

_____

B.3.19 Integration and Test - IM. The percentage of the adapted software's original integration and test effort expended.

_____.

B.3.20 Software Understanding - SU.

	Very Low	Low	Nom	High	Very High
Structure	Very low cohesion, high coupling, spaghetti code.	Moderately low cohesion, high coupling.	Reasonably well-structured; some weak areas.	High cohesion, low coupling.	Strong modularity, information hiding in data / control structures.
Application Clarity	No match between program and application world views.	Some correlation between program and application.	Moderate correlation between program and application.	Good correlation between program and application.	Clear match between program and application world-views.
Self Descriptiveness	Obscure code; documentation missing, obscure or obsolete	Some code commentary and headers; some useful documentation.	Moderate level of code commentary, headers, documentations.	Good code commentary and headers; useful documentation; some weak areas.	Self-descriptive code; documentation up-to-date, well-organized, with design rationale.
SU Increment to ESLOC	50	40	30	20	10

Table 20: Rating Scale for Software Understanding Increment SU

The *Software Understanding* increment (SU) is obtained from Table 20. SU is expressed quantitatively as a percentage. If the software is rated very high on structure, applications clarity, and self-descriptiveness, the software understanding and interface checking penalty is 10%. If the software is rated very low on these factors, the penalty is 50%. SU is determined by taking the subjective average of the three categories. Enter the percentage.

_____

B.3.21 Assessment and Assimilation - AA.

: Rating Scale for Assessment and Assimilation Increment (AA)

AA Increment	Level of AA Effort
0	None
2	Basic module search and documentation
4	Some module Test and Evaluation (T&E), documentation
6	Considerable module T&E, documentation
8	Extensive module T&E, documentation

The other nonlinear reuse increment deals with the degree of *Assessment and Assimilation* (AA) needed to determine whether a fully-reused software module is appropriate to the application, and to integrate its description into the overall

product description. Table provides the rating scale and values for the assessment and assimilation increment. Enter the percentage of AA:

_____

Post-Architecture Cost Drivers.

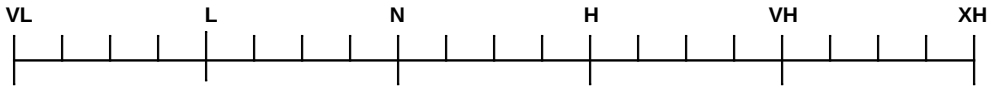
Use this section for **completed projects**. These are the 17 effort multipliers used in COCOMO II Post-Architecture model to adjust the nominal effort, Person Months, to reflect the software product under development. They are grouped into four categories: product, platform, personnel, and project. When an evaluation is in-between two rating levels always round to Nominal.

Product Cost Drivers.

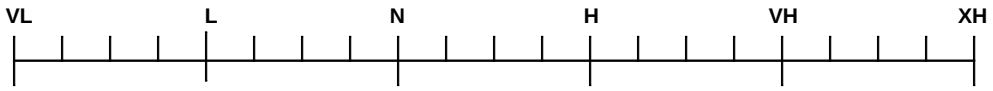
For maintenance projects, identify any differences between the base code and modified code Product Cost Drivers (e.g. complexity).

	Very Low	Low	Nominal	High	Very High	Extra High
RELY	slight inconvenience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
DATA		DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
RUSE		none	across project	across program	across product line	across multiple product lines
DOCU	Many life-cycle needs uncovered	Some life-cycle needs uncovered.	Right-sized to life-cycle needs	Excessive for life-cycle needs	Very excessive for life-cycle needs	

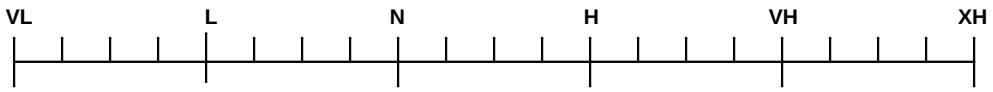
B.3.22 Required Software Reliability (RELY). This is the measure of the extent to which the software must perform its intended function over a period of time.



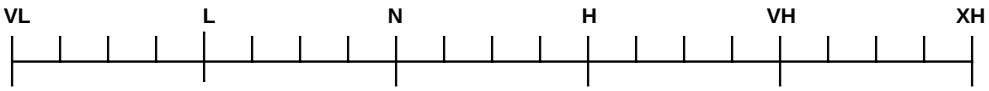
B.3.23 Data Base Size (DATA). This measure attempts to capture the affect large data requirements have on product development. The rating is determined by calculating D/P.



B.3.24 Required Reusability(RUSE). This cost driver accounts for the additional effort needed to construct components intended for reuse on the current or future projects.



B.3.25 Documentation match to life-cycle needs (DOCU). This captures the suitability of the project's documentation to its life-cycle needs. See the User's Manual.

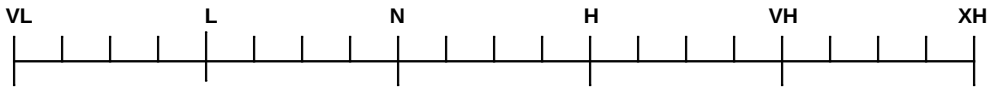


B.3.26 Product Complexity (CPLX):

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very Low	Straight-line code with a few non-nested structured programming operators: DOs, CASEs, IFTHENELSEs. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions: e.g., *A=B+C*(D-E)*	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTS-DB queries, updates.	Simple input forms, report generators.
Low	Straightforward nesting of structured programming operators. Mostly simple predicates	Evaluation of moderate-level expressions: e.g., *D=SQRT(B**2-4.*A*C)*	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Use of simple graphic user interface (GUI) builders.
Nominal	Mostly simple nesting. Some intermodule control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set.
High	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	Basic numerical analysis: multivariate interpolation, ordinary differential equations. Basic truncation, roundoff concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlap.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O, multimedia.

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very High	Reentrant and recursive coding. Fixed-priority interrupt handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/ 3D, dynamic graphics, multimedia.
Extra High	Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Distributed hard real-time control.	Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data. Complex parallelization.	Device timing-dependent coding, micro-programmed operations. Performance-critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality.

Complexity is divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations. Select the area or combination of areas that characterize the product or a sub-system of the product. The complexity rating is the subjective weighted average of these areas. The Post-Arch model only used one value for all 5 areas but for data collection purposes we are collecting the rating of each of the areas.

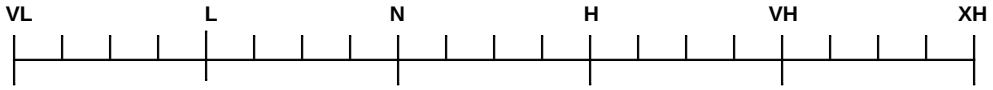


Platform Cost Drivers.

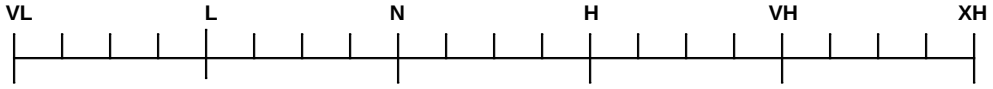
The platform refers to the target-machine complex of hardware and infrastructure software.

	Very Low	Low	Nominal	High	Very High	Extra High
TIME			≤ 50% use of available execution time	70%	85%	95%
STOR			≤ 50% use of available storage	70%	85%	95%
PVOL		major change every 12 mo.; minor change every 1 mo.	major: 6 mo.; minor: 2 wk.	major: 2 mo.; minor: 1 wk.	major: 2 wk.; minor: 2 days	

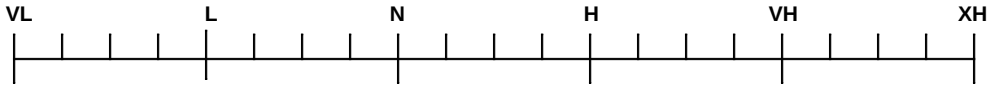
B.3.27 Execution Time Constraint (TIME). This is a measure of the execution time constraint imposed upon a software system.



B.3.28 Main Storage Constraint (STOR). This rating represents the degree of main storage constraint imposed on a software system or subsystem. See the User's Manual.



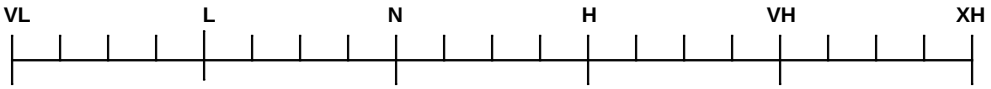
B.3.29 Platform Volatility (PVOL). "Platform" is used here to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks.



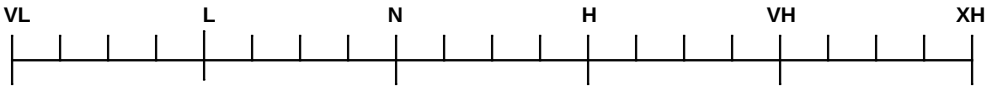
Personnel Cost Drivers.

	Very Low	Low	Nominal	High	Very High	Extra High
ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
PCON	48% / year	24% / year	12% / year	6% / year	3% / year	
AEXP	≤ 2 months	6 months	1 year	3 years	6 years	
PEXP	≤ 2 months	6 months	1 year	3 years	6 year	
LTEX	≤ 2 months	6 months	1 year	3 years	6 year	

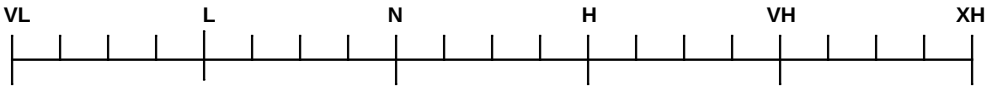
B.3.30 Analyst Capability (ACAP). Analysts are personnel that work on requirements, high level design and detailed design. See the User's Manual.



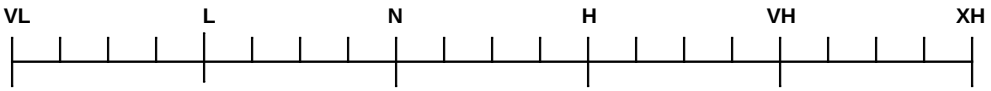
B.3.31 Programmer Capability (PCAP). Evaluation should be based on the capability of the programmers as a team rather than as individuals. Major factors which should be considered in the rating are ability, efficiency and thoroughness, and the ability to communicate and cooperate. See the User's Manual.



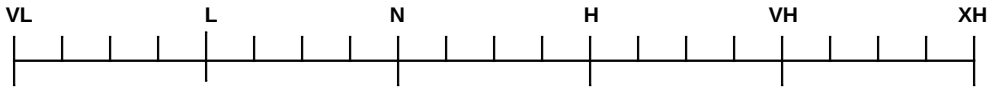
B.3.32 Applications Experience (AEXP). This rating is dependent on the level of applications experience of the project team developing the software system or subsystem. The ratings are defined in terms of the project team's equivalent level of experience with this type of application. See the User's Manual.



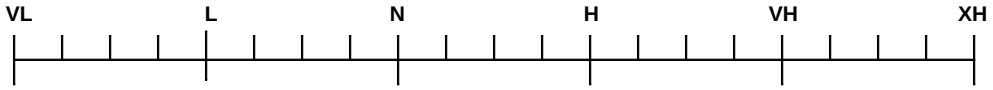
B.3.33 Platform Experience (PEXP). The Post-Architecture model broadens the productivity influence of PEXP, recognizing the importance of understanding the use of more powerful platforms, including more graphic user interface, database, networking, and distributed middleware capabilities. See the User's Manual.



B.3.34 Language and Tool Experience (LTEX). This is a measure of the level of programming language and software tool experience of the project team developing the software system or subsystem. See the User's Manual.



B.3.35 Personnel Continuity (PCON). The rating scale for PCON is in terms of the project's annual personnel turnover.

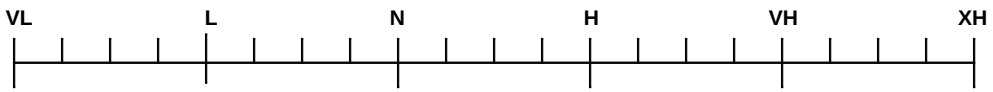


Project Cost Drivers.

This table gives a summary of the criteria used to select a rating level for project cost drivers.

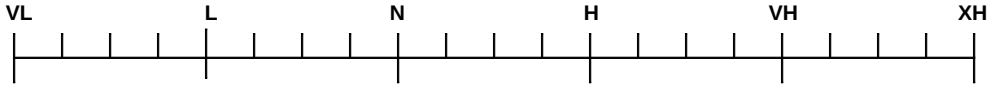
	Very Low	Low	Nominal	High	Very High	Extra High
TOOL	edit, code, debug	simple, front-end, backend CASE, little integration	basic lifecycle tools, moderately integrated	strong, mature lifecycle tools, moderately integrated	strong, mature, proactive lifecycle tools, well integrated with processes, methods, reuse	
SITE: Collocation	International	Multi-city and Multi-company	Multi-city or Multi-company	Same city or metro. area	Same building or complex	Fully collocated
SITE: Communications	Some phone, mail	Individual phone, FAX	Narrowband email	Wideband electronic communication.	Wideband elect. comm, occasional video conf.	Interactive multimedia
SCED	75% of nominal	85%	100%	130%	160%	

B.3.36 Use of Software Tools (TOOL). See the User's Manual.

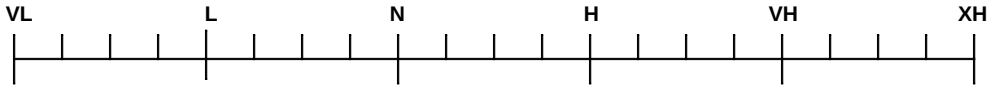


B.3.37 Multisite Development (SITE). Given the increasing frequency of multisite developments, and indications from Given the increasing frequency of multisite developments, and indications that multisite development effects are significant, the SITE cost driver has been added in COCOMO II. Determining its cost driver

rating involves the assessment and averaging of two factors: site collocation (from fully collocated to international distribution) and communication support (from surface mail and some phone access to full interactive multimedia). See the User's Manual.



B.3.38 Required Development Schedule (SCED). This rating measures the schedule constraint imposed on the project team developing the software. The ratings are defined in terms of the percentage of schedule stretch-out or acceleration with respect to a nominal schedule for a project requiring a given amount of effort. See the User's Manual.



Appendix C

DISTRIBUTION OF PREDICTOR AND KPA VARIABLES

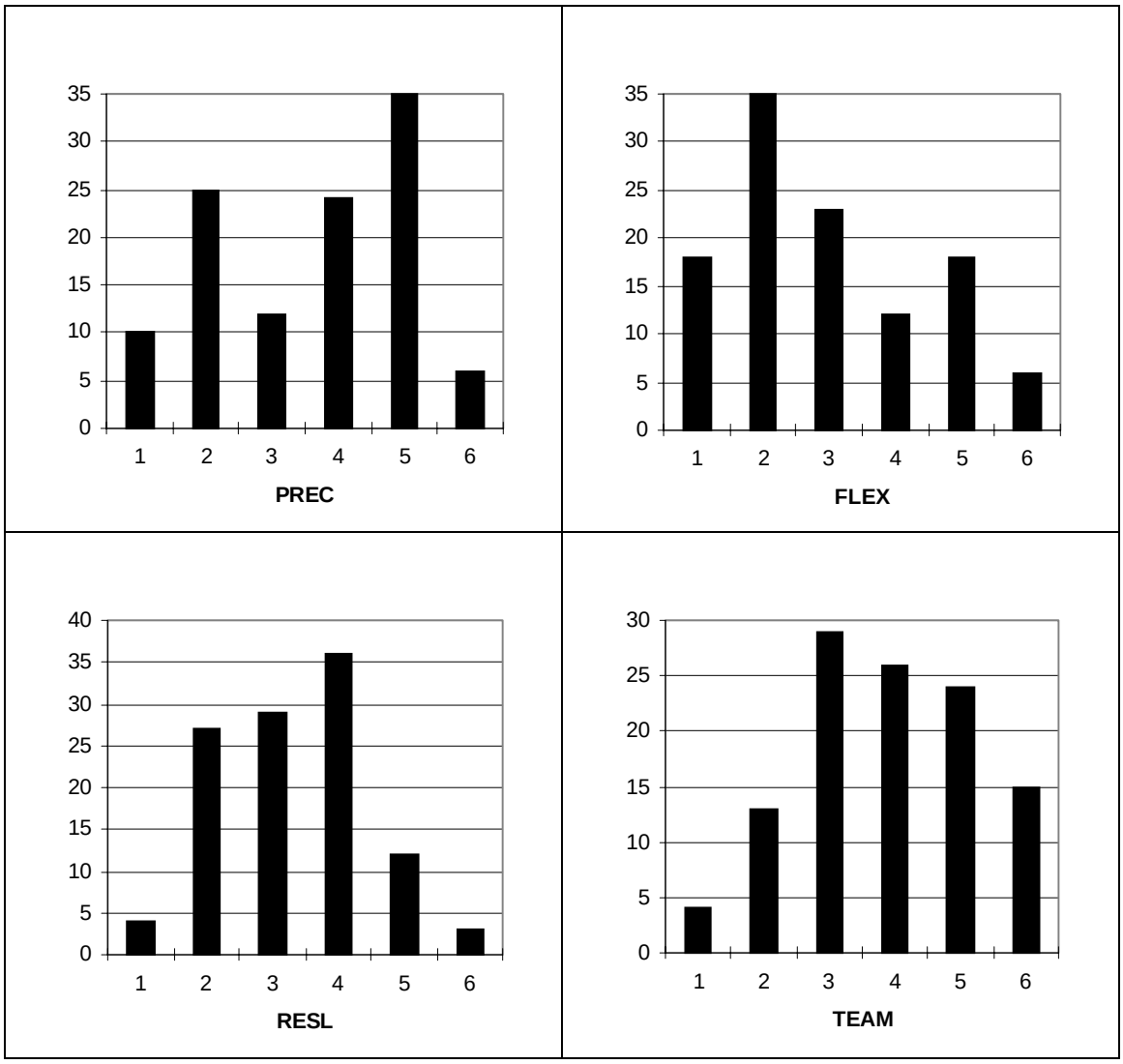
C.1 Predictor Distribution for 112 Observations

This data was created by assigning the values {1, 2, 3, 4, 5, 6} to the symbols {VL, L, N, H, VH, XH} respectively for each predictor variable in the data set. The ordinal values correspond with the R1, R2, R3, R4, R5, and R6 ratings in Table 14. Not all predictors have six valid; see section 5.4 on page 53.

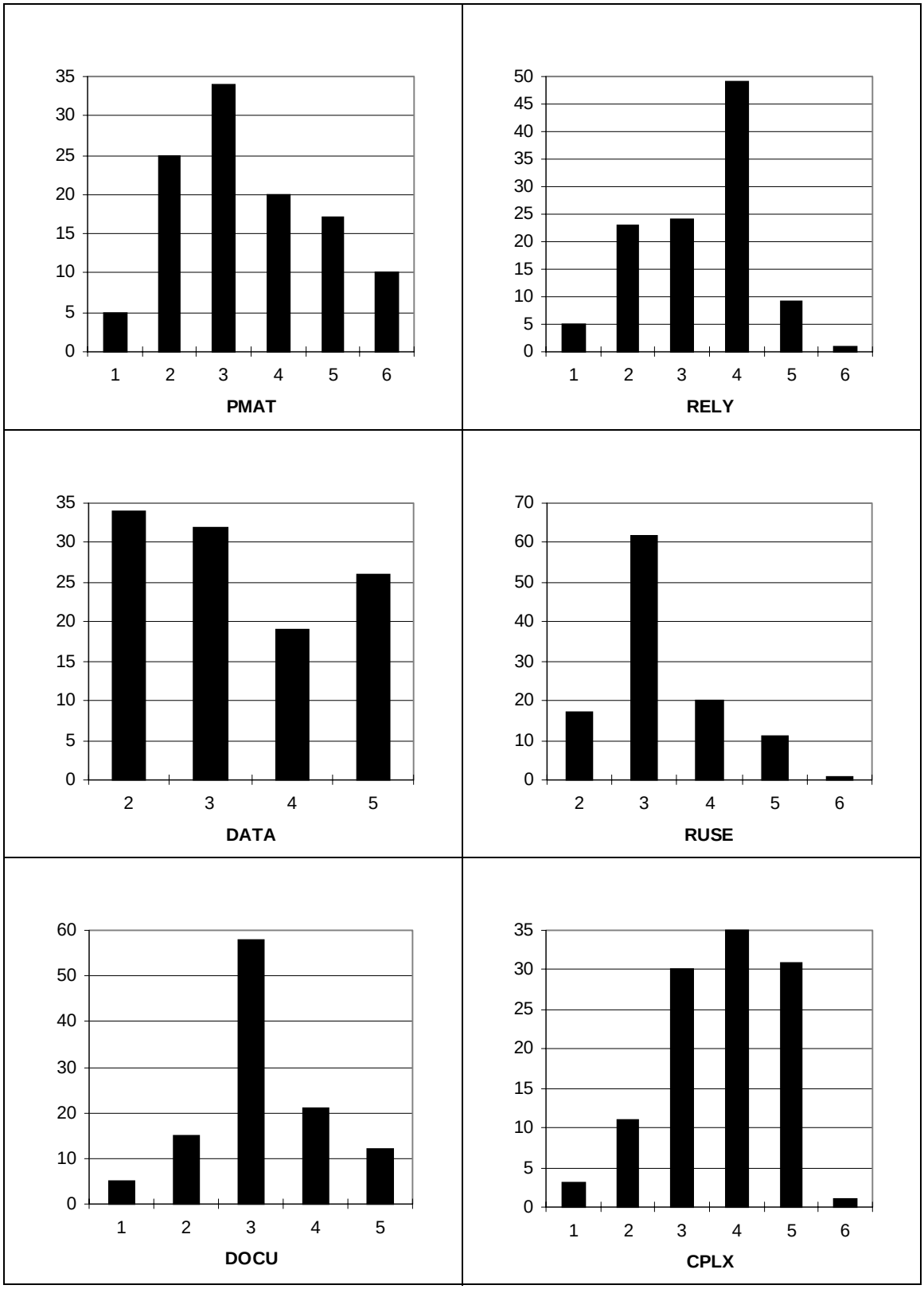
C.1.1 Predictor Summary Statistics[1]

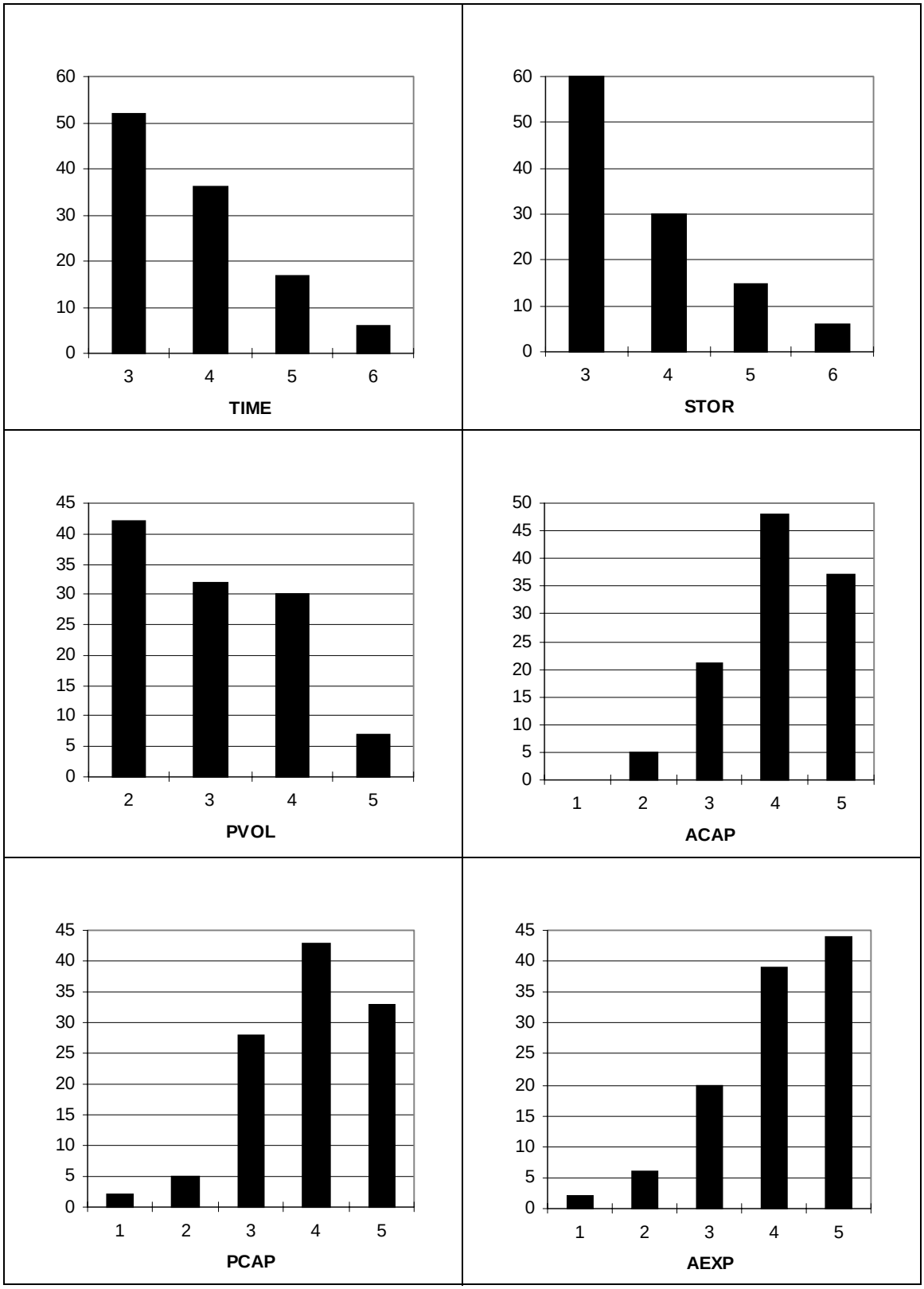
Variable	N	Average	Std. Dev	Minimum	Median	Maximum
KSLOC2	112	158.33	264.58	2.6	53.37	1264.
PREC	112	3.5469	1.4641	1.	4.	6.
FLEX	112	2.9129	1.4638	1.	3.	6.
RESL	112	3.2232	1.1162	1.	3.	6.
TEAM	112	3.8393	1.3372	1.	4.	6.
PMAT	112	3.219	1.2497	1.	3.	6.
RELY	112	3.2701	1.0527	1.	3.5	5.5
DATA	112	3.3058	1.1467	2.	3.	5.
RUSE	112	3.2277	0.84266	2.	3.	5.75
DOCU	112	3.1183	0.94735	1.	3.	5.
CPLX	112	3.6049	1.0292	1.	3.5	6.
TIME	112	3.7031	0.83594	3.	3.5	6.
STOR	112	3.6763	0.8819	2.5	3.	6.
PVOL	112	2.9844	0.93438	2.	3.	5.
ACAP	112	3.9665	0.79447	2.	4.	5.
PCAP	112	3.8058	0.90027	1.	4.	5.
AEXP	112	4.0022	0.99009	1.	4.	5.
PEXP	112	3.125	1.0349	1.	3.	5.
LTEX	112	3.1563	0.98418	1.	3.	5.
PCON	112	3.3839	0.84158	1.25	3.	5.
TOOL	112	1.9911	1.0424	1.	2.	5.
SITE	112	4.2902	1.0215	2.	4.	6.
SCED	112	2.7835	0.85061	1.	3.	5.

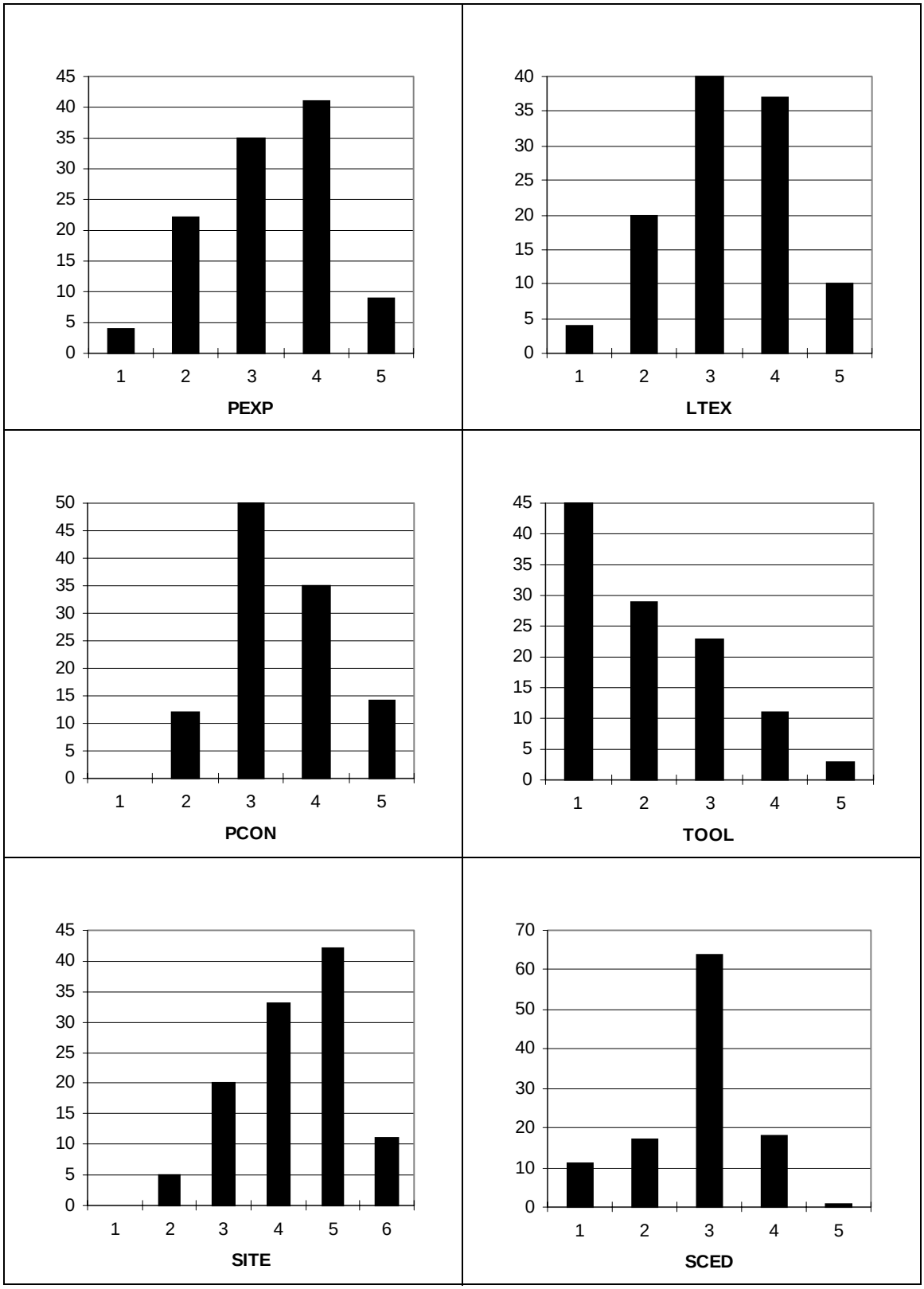
C.1.2 Histograms for each cost driver.



1. Data set = Db3_v10_1_Distribution


35
30
25
20
15
10
5
0
1 2 3 4 5 6
PMAT
50
45
40
35
30
25
20
15
10
5
0
1 2 3 4 5 6
RELY
35
30
25
20
15
10
5
0
2 3 4 5
DATA
70
60
50
40
30
20
10
0
2 3 4 5 6
RUSE
60
50
40
30
20
10
0
1 2 3 4 5
DOCU
35
30
25
20
15
10
5
0
1 2 3 4 5 6
CPLX


60
50
40
30
20
10
0
3
4
5
6
TIME
60
50
40
30
20
10
0
3
4
5
6
STOR
45
40
35
30
25
20
15
10
5
0
2
3
4
5
PVOL
50
45
40
35
30
25
20
15
10
5
0
1
2
3
4
5
ACAP
45
40
35
30
25
20
15
10
5
0
1
2
3
4
5
PCAP
45
40
35
30
25
20
15
10
5
0
1
2
3
4
5
AEXP


45
40
35
30
25
20
15
10
5
0
1 2 3 4 5
PEXP
40
35
30
25
20
15
10
5
0
1 2 3 4 5
LTEX
50
45
40
35
30
25
20
15
10
5
0
1 2 3 4 5
PCON
45
40
35
30
25
20
15
10
5
0
1 2 3 4 5
TOOL
45
40
35
30
25
20
15
10
5
0
1 2 3 4 5 6
SITE
70
60
50
40
30
20
10
0
1 2 3 4 5
SCED

C.1.3 Pairwise Correlations from the Data Set

KSLOC2	1.0000											
PREC	0.0656	1.0000										
FLEX	-0.0891	0.5376	1.0000									
RESL	-0.0314	0.3811	0.2942	1.0000								
TEAM	-0.1102	0.5164	0.5563	0.5829	1.0000							
PMAT	-0.0237	0.1330	-0.1638	0.2569	-0.0051	1.0000						
RELY	0.2328	-0.2238	-0.4998	-0.1505	-0.4157	0.1583	1.0000					
DATA	0.2786	0.1195	-0.1500	-0.0908	-0.0396	0.1982	0.0672	1.0000				
RUSE	-0.1340	-0.0786	-0.1541	0.1466	0.0078	0.2066	0.1471	-0.0838	1.0000			
DOCU	-0.0906	-0.1538	-0.3372	-0.0438	-0.2218	0.2032	0.2872	0.0410	0.1726	1.0000		
CPLX	0.1638	-0.0963	-0.1348	-0.0779	-0.0965	-0.0410	0.5094	-0.1958	-0.0284	0.2326	1.0000	
TIME	0.2067	-0.2563	-0.4511	-0.1830	-0.2979	-0.1065	0.5846	-0.0372	0.0616	0.1087	0.4593	1.0000
STOR	-0.0836	-0.2193	-0.3012	-0.0941	-0.1371	-0.3575	0.3558	-0.1785	0.0273	0.0220	0.2679	**0.6590**
	KSLOC2	PREC	FLEX	RESL	TEAM	PMAT	RELY	DATA	RUSE	DOCU	CPLX	TIME

STOR	1.0000										
PVOL	0.1865	1.0000									
ACAP	-0.0044	0.2147	1.0000								
PCAP	-0.0969	0.1925	**0.6694**	1.0000							
AEXP	0.1440	0.0883	0.2592	0.1881	1.0000						
PEXP	-0.2675	-0.4615	0.0517	0.1248	0.2349	1.0000					
LTEX	-0.3136	-0.4296	-0.0091	0.0466	-0.0009	**0.6490**	1.0000				
PCON	-0.1042	-0.0181	0.0758	0.1773	-0.1538	-0.0414	-0.0112	1.0000			
TOOL	-0.3903	0.1611	0.1968	0.2789	-0.1713	0.0934	0.1386	0.1278	1.0000		
SITE	-0.2848	-0.2052	0.1585	0.1953	-0.0574	0.3180	0.3236	0.2956	0.1584	1.0000	
SCED	-0.0928	-0.1127	-0.1591	-0.2444	-0.1158	-0.0809	-0.0359	0.0731	0.0232	0.1002	1.0000
	STOR	PVOL	ACAP	PCAP	AEXP	PEXP	LTEX	PCON	TOOL	SITE	SCED

C.2 KPA Data Distribution for 50 Observations

This data was were generated by assigning the values {100, 75, 50, 25, 1, 0} to the ratings {Almost Always, Frequently, About Half, Occasionally, Rarely if Ever, Don't Know or Does Not Apply} respectively. For KPAs 1 to 13, 50 observations were used. For KPAs 14 to 18, 40 observations were used

C.2.1 Summary Statistics[2]

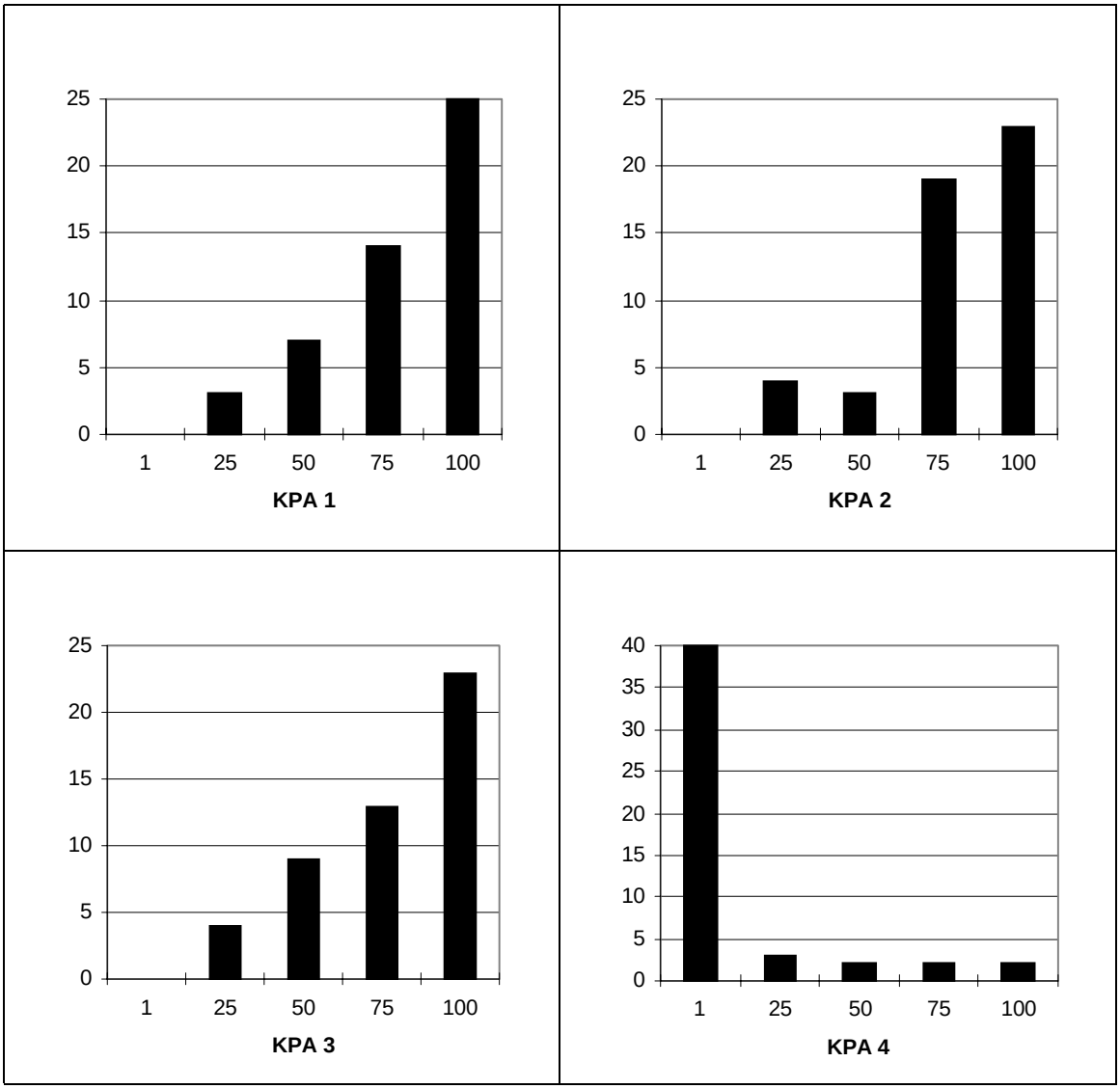
Variable	N	Average	Std. Dev.	Minimum	Median	Maximum
KPA1	50	80.4	23.164	25.	90.	100.
KPA2	50	80.7	23.168	25.	75.	100.
KPA3	50	77.5	25.158	25.	75.	100.
KPA4	50	10.22	25.103	0.	0.	100.
KPA5	50	73.3	27.004	0.	75.	100.
KPA6	50	82.9	17.026	50.	75.	100.
KPA7	50	64.16	37.014	1.	75.	100.
KPA8	50	69.82	33.848	1.	75.	100.
KPA9	50	55.96	32.796	0.	50.	100.
KPA10	50	64.84	35.26	1.	75.	100.
KPA11	50	75.22	23.451	1.	75.	100.
KPA12	50	68.8	25.982	0.	75.	100.
KPA13	50	60.28	36.245	0.	75.	100.

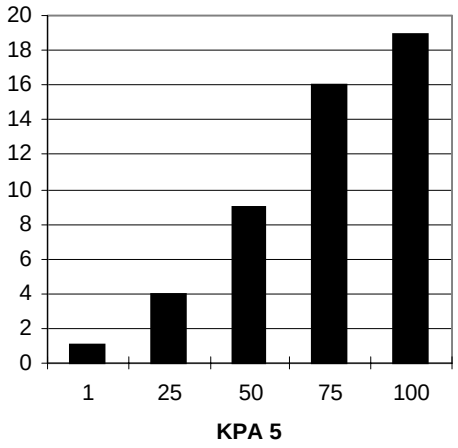
Variable	N	Average	Std. Dev.	Minimum	Median	Maximum
KPA14	40	47.025	35.238	0.	50.	100.
KPA15	40	47.275	34.468	0.	50.	100.
KPA16	40	42.325	36.129	1.	50.	100.
KPA17	40	37.875	38.442	0.	25.	100.
KPA18	40	46.4	38.785	1.	45.	100.

2. Data set = Db3_v10_1_Distribution_KPA

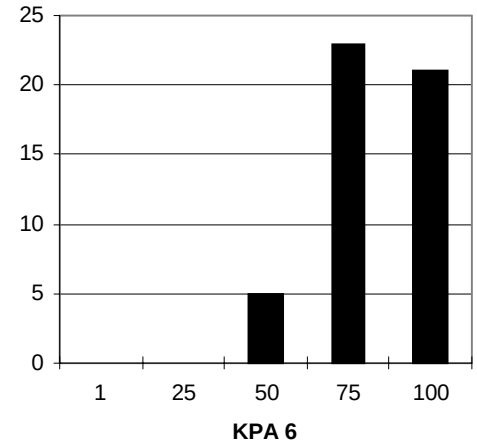
C.2.2 Histograms for each KPA

For KPAs 1 to 13, 50 observations were used to construct the histograms. For KPAs 14 to 18, 50 observations were used but 10 of those observations had 0 values.

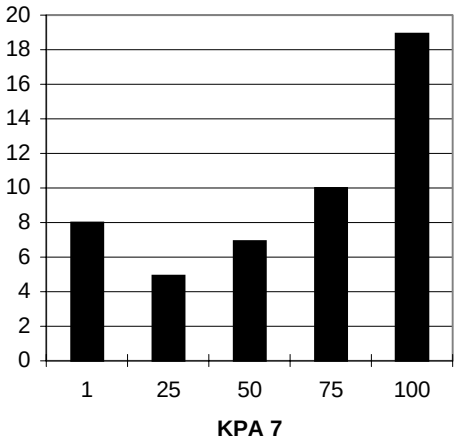




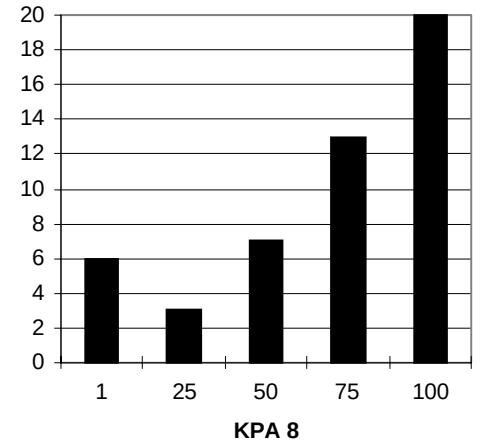
KPA 5



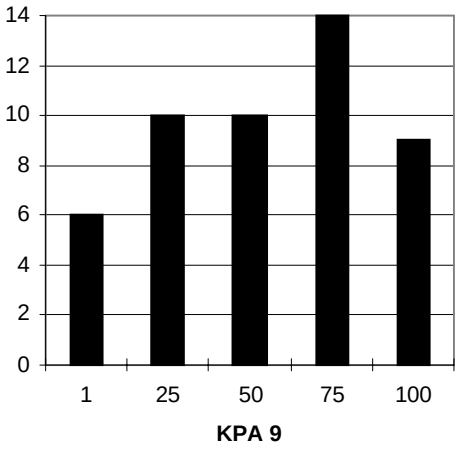
KPA 6



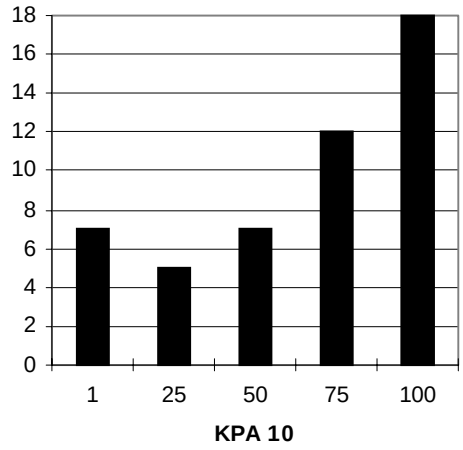
KPA 7



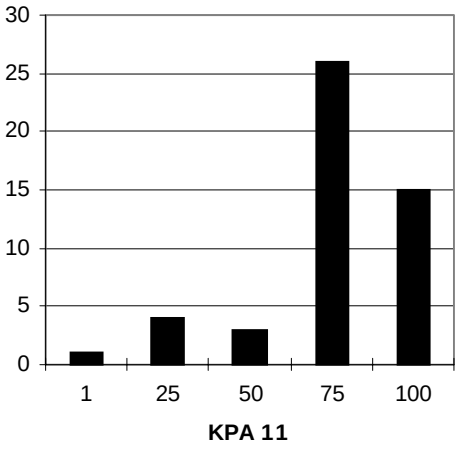
KPA 8



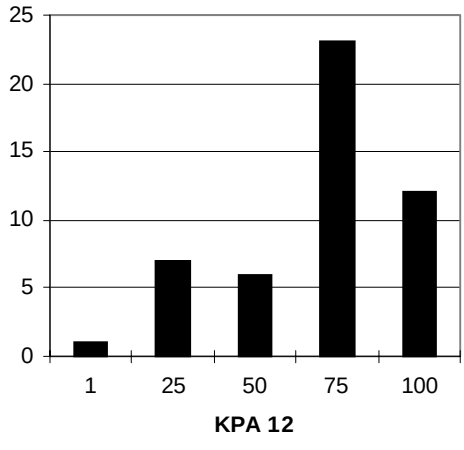
KPA 9



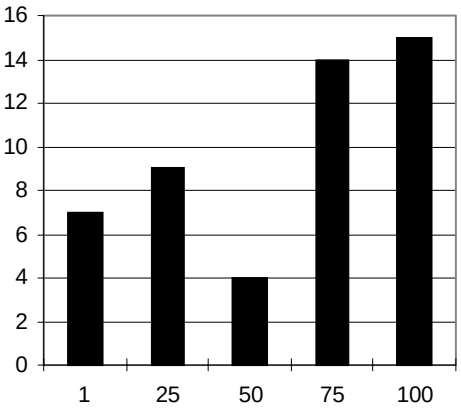
KPA 10



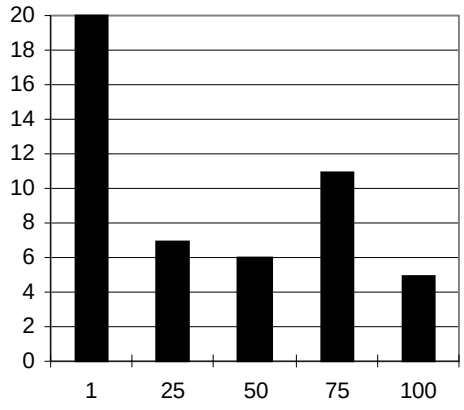
KPA 11



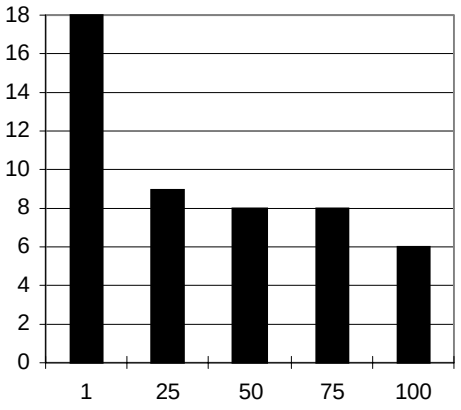
KPA 12



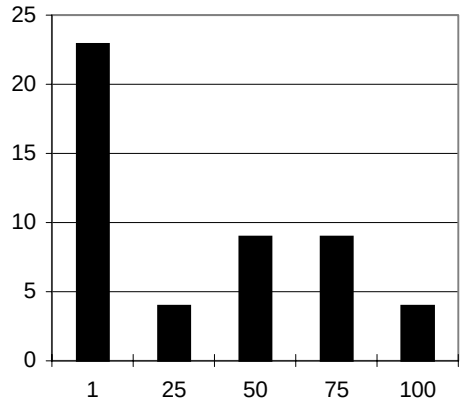
KPA 13



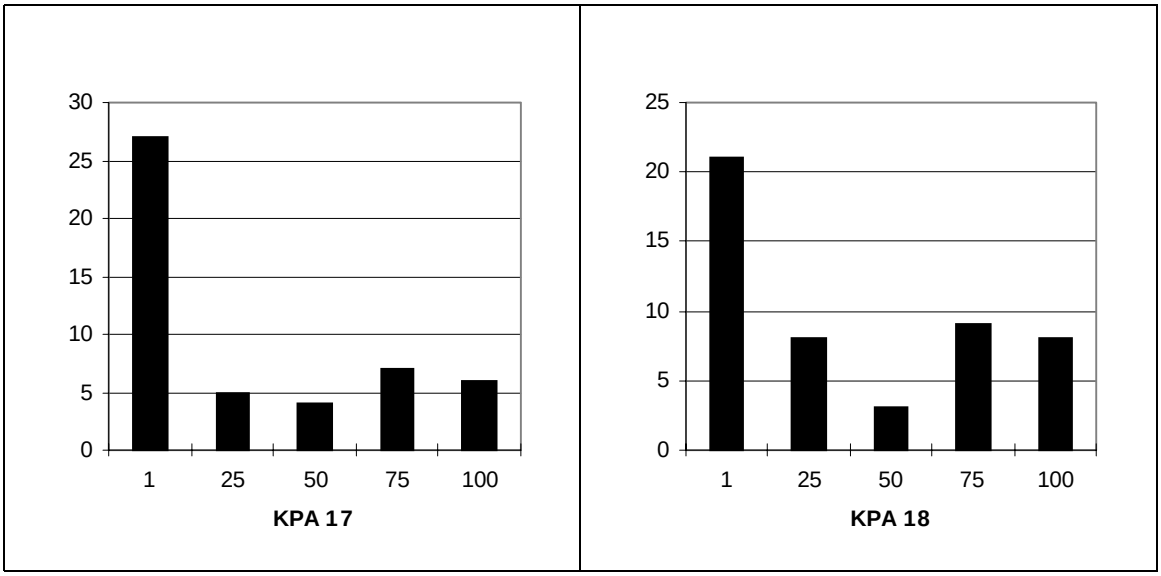
KPA 14



KPA 15



KPA 16


30
25
20
15
10
5
0
1
25
50
75
100
KPA 17
25
20
15
10
5
0
1
25
50
75
100
KPA 18

C.2.3 Pairwise Correlations from the Data Set

C.2.3.1 50 observations

KPA1	1.0000												
KPA2	0.5366	1.0000											
KPA3	0.6050	**0.8487**	1.0000										
KPA4	-0.1047	-0.2505	-0.2796	1.0000									
KPA5	0.5533	0.7302	**0.7259**	-0.1025	1.0000								
KPA6	0.5054	0.4526	0.5354	-0.0567	0.5458	1.0000							
KPA7	0.2608	0.5531	0.6060	0.0492	0.7030	0.4908	1.0000						
KPA8	0.2405	0.6084	0.6730	0.0564	0.6332	0.5098	**0.8456**	1.0000					
KPA9	0.4242	0.5135	0.5866	-0.1490	0.6091	0.3556	0.6734	0.5868	1.0000				
KPA10	0.2897	0.4933	0.6067	-0.0221	0.5650	0.5656	**0.7603**	**0.7240**	0.3998	1.0000			
KPA11	0.3725	0.3147	0.3519	-0.1448	0.4666	0.4032	0.3273	0.1782	0.3074	0.2181	1.0000		
KPA12	0.5544	0.3786	0.4808	-0.1246	0.3606	0.2860	0.3223	0.2586	0.4284	0.3765	0.4288	1.0000	
KPA13	0.3603	**0.7023**	**0.7507**	0.0100	0.6714	0.5189	**0.8241**	**0.7688**	0.5402	**0.8423**	0.2520	0.4526	1.0000
	KPA1	KPA2	KPA3	KPA4	KPA5	KPA6	KPA7	KPA8	KPA9	KPA10	KPA11	KPA12	KPA13

C.2.3.2 40 observations

KPA1	0.3656	0.1538	0.3524	0.3035	0.4065
KPA2	0.3963	0.3848	0.3186	0.4774	0.5152
KPA3	0.4127	0.3470	0.2681	0.4602	0.5301
KPA4	0.1880	0.1098	0.2361	-0.0168	0.1713
KPA5	0.3677	0.2999	0.4161	0.5255	0.5963
KPA6	0.3017	0.3983	0.2472	0.0875	0.3174
KPA7	0.2962	0.2303	0.4045	0.5288	0.6770
KPA8	0.3460	0.3956	0.4236	0.4748	**0.6191**
KPA9	0.3087	0.2134	0.4109	0.3980	0.5122
KPA10	0.1978	0.2354	0.0866	0.3024	0.4427
KPA11	0.2925	0.2369	0.2214	0.2113	0.3984
KPA12	0.2833	0.1698	0.1458	0.2352	0.4039
KPA13	0.3205	0.4110	0.2463	0.5070	0.5846
KPA14	1.0000	0.3537	0.3714	0.0012	0.3194
KPA15	0.3537	1.0000	0.5855	0.4159	0.5887
KPA16	0.3714	0.5855	1.0000	0.5247	**0.6799**
KPA17	0.0012	0.4159	0.5247	1.0000	**0.7492**
KPA18	0.3194	0.5887	**0.6799**	**0.7492**	1.0000
	KPA14	KPA15	KPA16	KPA17	KPA18

Appendix A

ANALYSIS RESULTS

D.1 Full Research Model - All

Using all of the observations, these are the results from using all of the predictor variables in the Research Model.

```
Data set = Db3_v11_970617
Response = log[EFFORT]
Coefficient Estimates:
Label         Estimate        Std. Error    t-value
Constant      0.796782        0.240585        3.312
log[KSLOC2]   1.04932         0.0442353      23.721
log[PMAT]     1.48913         0.663393        2.245
log[PREC]     0.474474        0.449612        1.055
log[RESL]    -0.227454        0.607055       -0.375
log[RELY]     0.257940        0.770396        0.335
log[DATA]     0.879129        0.486636        1.807
log[CPLX]     1.45251         0.643870        2.256
log[RUSE]    -0.263324        0.628031       -0.419
log[DOCU]     0.318624        0.596665        0.534
log[RCON]     3.54582         0.892315        3.974
log[PERS]     2.64321         0.847329        3.119
log[AEXP]    -0.391858        0.585143       -0.670
log[PEXP]     0.860007        0.733443        1.173
log[LTEX]    -0.470369        0.737439       -0.638
log[PCON]     0.215602        0.611967        0.352
log[TEAM]     0.483059        0.604500        0.799
log[FLEX]     0.165126        0.522200        0.316
log[TOOL]    -0.263195        0.655921       -0.401
log[SITE]     1.31893         0.661568        1.994
log[PVOL]     0.841042        0.699746        1.202
log[SCED]     3.10286         0.768474        4.038

R Squared:               0.935788
Sigma hat:               0.465415
Number of cases:             112
Degrees of freedom:           90

Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression     21   284.109        13.529      62.46    0.0000
Residual       90    19.495      0.216611
```

1.1 Full Research Model - Cross Validation

For cross-validation, these are the results for the full Model using only the calibration set of observations:

```
Data set = Db3_v11_970617_Xval
Response      = log[EFFORT]
Coefficient Estimates
Label         Estimate        Std. Error    t-value
Constant      0.919514        0.292618        3.142
log[KSLOC2]   1.03202         0.0552520      18.678
log[PMAT]     1.42425         0.788834        1.806
log[PREC]     0.667982        0.556857        1.200
log[RESL]    -0.630829        0.779058       -0.810
log[RELY]     1.05994         0.942694        1.124
log[DATA]     0.330507        0.579183        0.571
log[CPLX]     1.05988         0.746511        1.420
log[RUSE]    -0.742793        0.745860       -0.996
log[DOCU]     0.218062        0.722029        0.302
log[RCON]     4.29785         1.15614         3.717
log[PERS]     2.62630         0.948850        2.768
log[AEXP]    -1.35602         0.840944       -1.613
log[PEXP]     0.818338        0.851910        0.961
log[LTEX]     0.0428895       0.867368        0.049
log[PCON]     0.581018        0.746053        0.779
log[TEAM]     0.895094        0.741495        1.207
log[FLEX]    -0.351118        0.632725       -0.555
log[TOOL]    -0.776168        0.880840       -0.881
log[SITE]     0.939589        0.817975        1.149
log[PVOL]    -0.213260        0.825159       -0.258
log[SCED]     2.68509         0.886308        3.030
R Squared:               0.931845
Sigma hat:               0.474334
Number of cases:              84
Degrees of freedom:           62
Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression     21   190.723       9.08205      40.37    0.0000
Residual       62   13.9496      0.224993
```

1.2 Reduced Research Model - All

Using all of the observations, these are the results from pruning the Research Model down to ten predictors:

```
Data set = Db3_v11_970617
Response = log[EFFORT]
Coefficient Estimates:
Label         Estimate        Std. Error    t-value
Constant      0.786106        0.179891        4.370
log[KSLOC2]   1.05095         0.0369861      28.415
log[PMAT]     1.30885         0.485761        2.694
log[DATA]     0.862294        0.435838        1.978
log[CPLX]     1.43726         0.546981        2.628
log[RCON]     3.83335         0.653661        5.864
log[PERS]     2.38454         0.739814        3.223
log[TEAM]     0.879949        0.395178        2.227
log[SITE]     1.52240         0.566938        2.685
log[PVOL]     1.28229         0.558666        2.295
log[SCED]     3.09489         0.682231        4.536

R Squared:               0.933491
Sigma hat:               0.447128
Number of cases:             112
Degrees of freedom:          101

Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression     10   283.411       28.3411     141.76    0.0000
Residual      101   20.1923      0.199924
```

1.3 Reduced Research Model - Cross Validation

For cross-validation, these are the results for the ten variable Reduce Model using only the calibration set of observations:

```
Data set = Db3_v11_970617_Xval
Response      = log[EFFORT]
Coefficient Estimates:
Label         Estimate        Std. Error    t-value
Constant      0.790715        0.212977        3.713
log[KSLOC2]   1.04961         0.0444649      23.605
log[PMAT]     1.38333         0.590911        2.341
log[DATA]     0.413613        0.526307        0.786
log[CPLX]     1.39871         0.618259        2.262
log[RCON]     4.32553         0.800964        5.400
log[PERS]     1.90445         0.816951        2.331
log[TEAM]     1.02954         0.473071        2.176
log[SITE]     1.44455         0.641958        2.250
log[PVOL]     0.469202        0.672104        0.698
log[SCED]     2.56069         0.778700        3.288

R Squared:              0.925031
Sigma hat:              0.458467
Number of cases:             84
Degrees of freedom:          73

Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression     10   189.329       18.9329      90.07    0.0000
Residual       73    15.344      0.210192
```

1.4 Compact Research Model - All

Using all of the observations, the estimation for the compact Research Model is:

```
Data set = Db3_v11_970617
Response      = log[EFFORT]
Coefficient Estimates:
Label         Estimate        Std. Error    t-value
Constant      1.05511         0.165850        6.362
log[KSLOC2]   1.02607         0.0367183      27.944
log[PMAT]     2.02038         0.476405        4.241
log[PROD]     0.653098        0.152087        4.294
log[DEVT]     0.359646        0.160309        2.243
log[ENVR]     0.831887        0.276917        3.004

R Squared:               0.910937
Sigma hat:               0.505068
Number of cases:             112
Degrees of freedom:          106

Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression      5   276.564       55.3128     216.83    0.0000
Residual      106   27.0399      0.255094
```

1.5 Compact Research Model - Cross Validation

Using only the calibration observations the estimated coefficients for the five variable Compact Research Model is:

```
Data set = Db3_v11_970617_Xval
Response      = log[EFFORT]
Coefficient Estimates:
Label         Estimate        Std. Error    t-value
Constant      1.11876         0.203559        5.496
log[KSLOC2]   1.01072         0.0458399      22.049
log[PMAT]     2.43036         0.580460        4.187
log[PROD]     0.701924        0.194504        3.609
log[DEVT]     0.393635        0.196436        2.004
log[ENVR]     0.627138        0.330555        1.897

R Squared:                0.89916
Sigma hat:               0.514399
Number of cases:              84
Degrees of freedom:           78

Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression      5   184.033       36.8067     139.10    0.0000
Residual       78   20.6393      0.264606
```

1.6 Small Research Model - All

Using all of the observations, these are the results for estimating the three predictor Small Research Model:

```
Data set = Db3_v11_970617
Response = log[EFFORT]
Coefficient Estimates:
Label         Estimate        Std. Error    t-value
Constant      1.09681         0.157295        6.973
log[KSLOC2]   1.01986         0.0362210      28.157
log[PMAT]     2.11616         0.443367        4.773
log[EM]       0.636091        0.0671441       9.474

R Squared:               0.903052
Sigma hat:               0.522048
Number of cases:             112
Degrees of freedom:          108

Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression      3    274.17         91.39     335.33    0.0000
Residual      108   29.4337      0.272535
```

1.7 Small Research Model - Cross Validation

Using only the calibration set of observations, these are the results for estimating the three predictor Small Research Model:

```
Data set = Db3_v11_970617_Xval
Response      = log[EFFORT]
Coefficient Estimates:
Label         Estimate        Std. Error    t-value
Constant      1.11911         0.185702        6.026
log[KSLOC2]   1.01160         0.0436769      23.161
log[PMAT]     2.46227         0.537821        4.578
log[EM]       0.613062        0.0761062       8.055

R Squared:               0.891633
Sigma hat:               0.526541
Number of cases:             84
Degrees of freedom:          80

Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression      3   182.493        60.831     219.41    0.0000
Residual       80   22.1797      0.277246
```

1.8 Full COCOMO II Model - All

Using all the observations, the estimated Full COCOMO II model is:

```
Data set = Db3_v11_C2
Response      = log[EFFORT]
Coefficient Estimates:
Label       Estimate        Std. Error    t-value
Constant    1.08849         0.202421        5.377
log[SIZE]   0.811825        0.0823121       9.863
PREC_LNS    0.894581        0.918885        0.974
FLEX_LNS    1.25707         1.14759         1.095
RESL_LNS   -0.176452        1.34640        -0.131
TEAM_LNS    1.60058         1.37306         1.166
PMAT_LNS    4.22667         1.38671         3.048
log[RELY]   0.822193        0.460910        1.784
log[DATA]   0.794800        0.684728        1.161
log[RUSE]  -0.318157        0.432453       -0.736
log[DOCU]  -0.0172566       0.766769       -0.023
log[CPLX]   1.26428         0.460036        2.748
log[RCON]   1.71444         0.548171        3.128
log[PVOL]   0.455017        0.493255        0.922
log[PERS]   1.87018         0.460419        4.062
log[AEXP]  -0.404470        0.530952       -0.762
log[PEXP]   1.26537         0.606701        2.086
log[LTEX]  -0.708010        0.679353       -1.042
log[PCON]  -0.138229        0.615606       -0.225
log[TOOL]  -0.137238        0.601720       -0.228
log[SITE]   0.324189        0.840655        0.386
log[SCED]   2.64426         0.767024        3.447

R Squared:               0.933373
Sigma hat:               0.474085
Number of cases:              112
Degrees of freedom:            90

Summary Analysis of Variance Table
Source         df       SS             MS           F    p-value
Regression     21   283.376       13.4941      60.04    0.0000
Residual       90   20.2281      0.224757
```

1.9 Full COCOMO II Model - Cross Validation

Using only the calibration observations, the estimated coefficients for the full COCOMO II model is:

```
Data set = Db3_v11_C2_Xval, Name of Model = L2
Response      = log[EFFORT]
Coefficient Estimates
Label       Estimate        Std. Error    t-value
Constant    0.844182        0.245004        3.446
log[SIZE]   0.944874        0.107182        8.816
PREC_LNS    0.883160        1.13329         0.779
FLEX_LNS    0.438326        1.38776         0.316
RESL_LNS   -1.69034         1.76402        -0.958
TEAM_LNS    1.58888         1.62094         0.980
PMAT_LNS    3.64052         1.66106         2.192
log[RELY]   0.533289        0.613838        0.869
log[DATA]   0.465244        0.774043        0.601
log[RUSE]  -0.550270        0.519236       -1.060
log[DOCU]   0.518967        0.885174        0.586
log[CPLX]   1.18649         0.508268        2.334
log[RCON]   2.92304         0.805223        3.630
log[PVOL]  -0.131208        0.566377       -0.232
log[PERS]   1.79118         0.512331        3.496
log[AEXP]  -0.613232        0.676852       -0.906
log[PEXP]   0.914033        0.703826        1.299
log[LTEX]  -0.299041        0.782096       -0.382
log[PCON]  -0.0357251       0.733913       -0.049
log[TOOL]  -0.308430        0.777198       -0.397
log[SITE]   0.887400        1.00075         0.887
log[SCED]   2.57817         0.862417        2.989

R Squared:               0.933985
Sigma hat:               0.473875
Number of cases:              84
Degrees of freedom:           62

Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression     21   196.976       9.37983      41.77    0.0000
Residual       62   13.9226      0.224557
```

1.10 Reduced COCOMO II Model - All

Using all of the observations, the estimated coefficients for the ten predictor Reduced COCOMO II model is:

```
Data set = Db3_v11_C2
Response      = log[EFFORT]
Coefficient Estimates:
Label       Estimate        Std. Error    t-value
Constant    1.01863         0.168203        6.056
log[SIZE]   0.892034        0.0639966      13.939
TEAM_LNS    2.64278         0.999088        2.645
PMAT_LNS    3.55860         1.08989         3.265
log[DATA]   0.984160        0.630235        1.562
log[CPLX]   1.39620         0.407362        3.427
log[RCON]   2.23342         0.467444        4.778
log[PVOL]   0.920725        0.405546        2.270
log[PERS]   1.72001         0.417995        4.115
log[SCED]   2.91359         0.698897        4.169
log[SITE]   1.13320         0.734880        1.542

R Squared:               0.925634
Sigma hat:               0.472802
Number of cases:             112
Degrees of freedom:          101

Summary Analysis of Variance Table
Source         df       SS            MS           F    p-value
Regression     10   281.026       28.1026     125.72    0.0000
Residual      101   22.5777      0.223542
```

1.11 Reduced COCOMO II Model - Cross Validation

Using only the calibration observations, the estimated coefficients for the ten variable Reduced COCOMO II model is:

```
Data set = Dv3_v11_C2_Xval
Response      = log[EFFORT]
Coefficient Estimates:
Label       Estimate        Std. Error    t-value
Constant    0.802689        0.184615        4.348
log[SIZE]   0.944137        0.0733542      12.871
TEAM_LNS    1.96491         1.09856         1.789
PMAT_LNS    3.22736         1.28858         2.505
log[DATA]   0.384699        0.707824        0.543
log[CPLX]   1.24893         0.432262        2.889
log[RCON]   3.36296         0.595562        5.647
log[PVOL]   0.257121        0.463105        0.555
log[PERS]   1.48057         0.441727        3.352
log[SITE]   1.13605         0.788598        1.441
log[SCED]   2.55000         0.756120        3.372

R Squared:               0.928048
Sigma hat:               0.455928
Number of cases:              84
Degrees of freedom:           73

Summary Analysis of Variance Table
Source         df       SS             MS           F     p-value
Regression     10   195.725       19.5725      94.16    0.0000
Residual       73   15.1745       0.20787
```