

THE EFFECTS OF CASE TOOLS ON SOFTWARE
DEVELOPMENT EFFORT

by

Jongmoon Baik

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)

December 2000

DEDICATION

To my wife and children, Youngmin Kim, Lucy Baik, and Christopher Baik

To my parents, Keumsun Baik and Soonja Kim

To my parents-in-law, Dalhong Kim and Jungae Jung

ACKNOWLEDGEMENTS

Even if this dissertation was dedicated to my family, parents, and parents-in-law, I would once again like to give special thanks to my wife, Youngmin Kim, for her great support, encouragement, endurance.

I would like to thank my advisor, Dr. Barry Boehm, for his excellent mentoring of me throughout the last four years. Without his support and guidance throughout the years, I would have not been able to continue my Ph.D studies.

I would also like to thank my committee members, Dr. Bert Steece and Dr. Nenad Medvidovic. I really appreciate Dr. Steece's deep involvement in my research as a outside member by providing me with statistical methodologies that are useful to build and to validate the research model. Dr. Nenad Medvidovic also gave me valuable advice for this research.

Dr. Ellis Horowitz must be also acknowledged for giving me a chance to work at the Center for Software Engineering and giving me valuable advice necessary for me to live both in academic world and outside world.

I would also like to thank all friends in the Center for Software Engineering and COCOMO II research group. I will not forget all the memories I had with them. I will miss all of them with a great deal.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
ABSTRACT.....	viii
1.0 Introduction.....	1
2.0 Software Cost Models and Productivity Impact	5
2.1 CASE Impact in Software Estimation Models	6
2.1.1 SLIM (Software Life Cycle Model).....	6
2.1.2 Jensen Model.....	9
2.1.3 Checkpoint	12
2.1.4 PRICE-S	15
2.1.5 Softcost.....	18
2.1.6 COCOMO	20
2.1.7 Overall Summary of Cost Estimation Models	24
2.2 Tool Impact on Software Productivity	25
3.0 COCOMO II and CASE Classification	28
3.1 COCOMO II and Its Database	28
3.1.1 COCOMO II Post Architecture Model	29
3.1.2 Analysis of the COCOMO II Database.....	37
3.2 Software Tools Classification by Lifecycle Activity	43
4.0 Research Approach	52
4.1 COCOMO II 7-Step Modeling Methodology	52
4.2 Step 1. Tool Rating Scales by Productivity Dimensions.....	55
4.2.1 Completeness of Tool Activity Coverage (TCOV).....	56
4.2.2 Degree of Tool Integration (TINT)	59
4.2.3 Tool Maturity and User Support (TMAT).....	61
4.3 Step 1. Research Model for TOOL Rating Dimensions	62
4.3.1 Multiple Regression Analysis	62
4.3.2 Research Model.....	66
4.4 Hypothesis Testing	70
4.5 Step 2 & 3. Behavioral Analysis: Project Activity Differences	71
5.0 Bayesian Approach for Model Calibration	74
5.1 Bayesian Approach	74
5.1.1 Step 4. Delphi Process Result (A-Priori Model)	74

5.1.2	Step 5. Multiple Regression Analysis (Sample).....	76
5.2	Step 6. Bayesian Analysis (A-Posteriori Model).....	79
6.0	Research Model Validation.....	85
6.1	Cross-Validation by Data-Splitting.....	85
6.1.1	Data Splitting.....	85
6.1.2	Cross-Validation in Arc.....	86
6.1.3	Cross-Validation Results Using Arc.....	87
6.2	Cross-Validation by Bootstrap.....	95
6.2.1	Bootstrap.....	95
6.2.2	Bootstrap in Arc.....	98
6.2.3	Bootstrap Results Using Arc.....	99
6.3	Summary of Cross Validation.....	108
7.0	Conclusions.....	110
7.1	Summary of Contributions.....	110
7.2	Future Research Directions.....	111
8.0	Bibliography.....	113
	Appendix A.Delphi Questionnaire.....	117
	Appendix B:Bootstrap Histograms and Probaility Plots.....	125
B.1	Histograms (One dimensional TOOL).....	125
B.2	Probability Plots (One dimensional TOOL).....	130
B.3	Histograms (Three dimensional TOOL).....	135
B.4	Probability Plots (Three dimensional TOOL).....	140

LIST OF TABLES

Table 2.1. SLIM Software Equation	7
Table 2.2. Constant Adjustment Factor	11
Table 2.3. Example: Adjustment Assumption of Macro estimation	13
Table 2.4. Example: Adjustment Assumptions for Microestimation	14
Table 2.5. CASE Integration Criteria	15
Table 2.6. PRICE-S Core Computation Flow	16
Table 2.7. Complexity Adjustment Factors	18
Table 2.8. Use of Software Tools and Software Tool/Environment Stability Matrix.....	20
Table 2.10. TOOL Rating in COCOMO and Ada COCOMO	23
Table 2.11. Summary of Cost Estimation Models.....	25
Table 2.12. Impact of Software Tools on Productivity	27
Table 3.1. Scale Factor Ratings of the COCOMO II.....	32
Table 3.2. Post Architecture Effort Multipliers	34
Table 3.3. Mean and Median Values for Size and Effort.....	39
Table 3.4. Classes of CASE Tools	47
Table 4.2. Rating Scale for Degree of Tool Integration.....	60
Table 4.3. Rating Scale for Tool Maturity and User Support	62
Table 4.4. Rating Summary for Effort Multipliers	67
Table 4.5. Complexity Ratings	68
Table 4.6. Project Activity Differences Due to Completeness of Activity Coverage	72
Table 4.7. Project Activity Differences Due to Degree of Tool Integration.....	73
Table 4.8. Project Activity Differences Due to Tool Maturity and User Support	73
Table 5.1. Initial A-Priori Weighting Values	75
Table 5.2. Final A-Priori Weighting Values	76
Table 5.3. Three Rating Scale Values from the data of 15 Projects	76
Table 5.4. TOOL conversion from COCOMO 81 to COCOMO II	77
Table 5.5. Regression Summary of the Sample.....	78
Table 5.6. Bayesian Analysis Procedure in Arc	82
Table 5.7. Point Estimates of Coefficients	83
Table 5.8. Prediction Accuracy Comparison	84
Table 6.1. Fit for the construction of 146 project data	89
Table 6.2. Cross Validation Summary and PRESS.....	90
Table 6.3. Fit for the construction set using one dimensional TOOL	92
Table 6.4. Cross Validation Summary and PRESS for one dimensional TOOL validation set.....	93
Table 6.5. Fit for the construction set using three dimensional TOOL	94
Table 6.6. Cross Validation Summary and PRESS for three dimensional TOOL validation set	95
Table 6.7. Fit of 161 project data (One-dimensional TOOL).....	100
Table 6.8. Fit of 161 project data (Three-dimensional TOOL)	104
Table 6.9. Bootstrap standard-error and bias	106
Table 6.10. Percentile bootstrap confidence intervals	107

LIST OF FIGURES

Figure 2.1. Basic Technology Constant.....	10
Figure 2.2. Jensen Software Development Effort Ratio.....	12
Figure 2.3. Productivity Improvement in a lifecycle phases.....	26
Figure 2.4. Degree of facilitating project tasks (indicating higher productivity).....	27
Figure 3.1. The Future of Software Market Place.....	29
Figure 3.2. Distribution of Size and Actual Effort for the COCOMO II.1998 161 dataset.....	38
Figure 3.3. Distribution of the COCOMO II TOOL Cost Driver.....	39
Figure 3.4. A-Posteriori Update for TOOL Rating Productivity Range.....	41
Figure 3.5. COCOMOII.1998 Productivity Ranges.....	41
Figure 3.6. Magnitude of Correlations with TOOL.....	42
Figure 3.7. Sommerville's Activity-based classification.....	44
Figure 3.8. Development tasks in Forte and McCully's classification.....	45
Figure 3.9. Activity Coverage of Software Tools.....	51
Figure 4.1. The seven-step modeling methodology.....	53
Figure 4.2. Tool-use characteristics at CMM Levels.....	57
Figure 5.1. Bayesian: Combination of Prior and Sampling Information.....	80
Figure 6.1. Histograms for Intercept and log[TOOL].....	102
Figure 6.2. Probability Plots for Intercept and log[TOOL].....	103
Figure 6.3. Percentile bootstrap confidence intervals for log[TOOL].....	108

ABSTRACT

CASE (Computer Aided Software Engineering) tools have played a critical role in improving software productivity and quality by assisting tasks in software development processes since the 1970's. Several parametric software cost models adopt "use of software tools" as one of the environmental factors that affect software development productivity. However, most software development teams use CASE tools that are assembled over time and adopt new tools without establishing formal evaluation criteria. Several software cost models assess the productivity impacts of CASE tools based just on breadth of tool coverage without considering other productivity dimensions such as degree of integration, tool maturity, and user support. This dissertation provides an extended set of tool rating scales based on the completeness of tool coverage, the degree of tool integration, and tool maturity/user support. It uses these scales to refine the way in which CASE tools are effectively evaluated within COCOMO (CONstructive COSt MOdel) II. In order to find a best fit of weighting values for the extended set of tool rating scales in the extended research model, a Bayesian approach is adopted to combine two sources of (expert-judged and data-determined) information to increase prediction accuracy. The research model using the extended three TOOL rating scales is validated by using cross-validation methodologies such as data splitting and bootstrapping.

1.0 Introduction

It is common knowledge that CASE (Computer Aided Software Engineering) tools have played a critical role in the software engineering process by improving software quality and productivity. A huge number of CASE tools have been produced to assist tasks in software development processes since the end of the 1970's. Many initiatives in the CASE field were pursued in the 1980's and the early 1990's to provide more effective CASE technologies and environments. Even though the research in this field is no longer as active, software development teams use a range of CASE tools that are typically assembled over the period to support tasks throughout the software production lifecycle. The diversity and proliferation of software tools in the current CASE market makes it difficult to understand what kind of tasks are supported and how much effort can be reduced by using CASE tools in software development processes. A big challenge is to alleviate these difficulties in the software engineering community. The primary goals of this dissertation are to establish a framework for classifying CASE tools according to the breadth of their support in a software production lifecycle, to provide rating scales with which CASE tools can be effectively evaluated, and to analyze the effect of software tools on the software development effort.

Some classifications of software tools (Forte, G. & McCully, K., 1991; Roger S. Pressman, 1992; Alfonso Fuggetta, 1993; Ian Sommerville, 1995) have been made to understand what tasks are supported by software tools in a software production process. Sommerville suggests three dimensions such as *Functionality*, *Process Support*, and

Breadth of Support with which software tools can be classified (Ian Sommerville, 1995). However, most of the classifications are based on the dimensions of *Process Support and Functionality* in the phases of a software development lifecycle. Thus, they do not account for the mutual dependencies and relationships among CASE tools in an entire software development lifecycle. Software tools can support one or more phases in the form of *Workbenches* or *Environments* (Alfonso Fuggetta, 1993). Fuggetta's classification of *Workbenches* does not show what degree of activities in the phases are covered. This dissertation provides a classification of CASE tools and technologies based on the coverage of activities supported in a software lifecycle.

Most software development teams use a variety of CASE tools with the hope of delivering-on-time. Ed Yourdon pointed out the importance of CASE tools in a software development process by suggesting the notion of a triage: “must have”, “should have”, and “could have” to avoid a *Death-March* project (Ed Yourdon, 1996). However, most software project teams use CASE tools that are assembled over time and adopt new CASE tools without establishing a formal evaluation criteria. Several researches (Robert Firth et al., 1987; Vicky Mosley 1992) provide guidelines for the assessment of CASE tools and several parametric software cost estimation models make “use of software tools” as one of the environmental factors in order to assess the impact of CASE tools on the software productivity.

During the last two decades, many software cost estimation models have been developed to provide estimates of effort and schedule that are used to manage software development processes. While most of those models are proprietary, COCOMO (Boehm,

B. W., 1981) is one of the available and widely accepted models in public and has been updated to the COCOMO II to address the issues such as non-sequential and rapid development models; reuse-driven approaches including commercial-off-the-shelf (COTS) packages, reengineering, application composition, and application generation capabilities; object-oriented approaches supported by distributed middleware; and software process maturity effects (Boehm, B. W. et al., 2000). The COCOMO II uses TOOL as one of effort multipliers to explain the impact on effort estimates and provides a TOOL rating scale that gives a guideline to assess software tools. The latest version of the COCOMO II gives fairly good prediction rates that are within 20% of the actuals 56% of the time, within 25% of the actuals 65% of the time, and within 30% of the actuals 71% of the time (Sunita Chulani, Boehm, B. W., & Bert Steece, 1999). Therefore, this research is based on the COCOMO II model.

Even if the COCOMO II TOOL rating scale was updated with available tools in the 1990's CASE market, it may not be complete without considering other factors such as tool integration, tool maturity and user support. Since the COCOMO II TOOL rating scale is too simple and gives ambiguous definitions of tools, it is difficult to fit a set of tools to an equivalent level of a CASE tool set. Much of tool evaluation criteria (Ian Sommerville, 1995; Ovum, 1995; Vicky Mosley 1992) considers the degree of tool integration, tool maturity in the market, and services provided by vendors, which are very important factors in software development processes to assess CASE tools in addition to the quality of tools. For these reasons, this dissertation provides the extended three dimensional rating scales such as *Completeness of Tool Coverage (TCOV)*, *Degree of Tool Integration*

(TINT), and *Tool Maturity/User Support* (TMAT) from the one dimensional COCOMO II TOOL rating scale.

The result of the COCOMO II Bayesian analysis (Sunita Chulani, Boehm, B. W., & Bert Steece, 1999) on data from 161 projects collected from USC-CSE (The Center for Software Engineering at University of Southern California) affiliates such as Commercial, Aerospace, Government, and FFRDC (Federally Funded Research and Development Centers) shows that the TOOL variable is statistically significant in predicting effort estimates. However, the Bayesian analysis focuses on only completeness of tool activity coverage like other software cost estimation models. Hence, this dissertation provides a mathematical model based on the 7-step COCOMO II modeling methodology to effectively identify the relative impact of the extended three dimensional TOOL rating scales on a project's effort estimation.

In practice, it is very difficult to get newly collected project data to validate a regression model. Thus, this dissertation adopts two cross validation methods (cross-validation by data splitting and Bootstrap methods) which are widely used in statistics. Thereby, this dissertation shows how to use those methods to validate a regression model and depicts the simulation results (with and without the three TOOL rating scales) of the two cross validation methods.

2.0 Software Cost Models and Productivity Impact

Appropriate use of CASE tools has led to the improvement of productivity and quality by replacing human tasks in a software lifecycle with automated processes since the end of the 1970's. In spite of the availability of a huge number of CASE tools, there is no widely-accepted, systematic, and formal set of evaluation criteria to appraise their usefulness for the improvement of productivity and quality. Many software estimation models adopt the "use of software tools" as one of the project factors which accounts for the development environment to predict its impact on development effort and schedule. COCOMO (COConstructive COSt MOdel) uses TOOL as one of the effort multipliers which accounts for differences in project effort (Boehm, B. W., 1981). Since COCOMO provides a categorization of the typical outdated set of tools available at each level, it requires some amount of judgement to fit a mixture of tools to an equivalent level of tool support. Also, it does not consider important factors such as the degree of tool integration, the maturity of software tools, and user support. COCOMO has been updated to Ada COCOMO (Boehm, B. W. & Walker Royce, 1989) and the COCOMO II (Boehm, B. W. et al., 2000). This chapter compares some popular, existing software estimation models to see how they treat the "use of software tools" in order to assess the impact on software development effort. It also shows the impacts of CASE tools on software development productivity in much of the literature.

2.1 CASE Impact in Software Estimation Models

During the last two decades, many software estimation models have been developed to accurately predict the cost of a software project. Many of them use the proprietary information to drive their estimation models. Therefore, they can not be compared against one another in terms of their model structure and how they deal with CASE tools to estimate the productivity impact on software development effort and schedule estimations. This section discusses a few popular estimation models focusing on CASE tools by examining how much CASE tools have an effect on the estimation of software development effort.

2.1.1 SLIM (Software Life Cycle Model)

SLIM (Software Life Cycle Model) is a quantitative and mathematical software estimation model developed by Larry Putnam in the late 70's (Lawrence H. Putnam & Ware Myers, 1992). It uses a negative exponential curve called the Rayleigh-Norden effort curve, which indicates the man-power distribution over time during the software lifecycle. SLIM has been enhanced by Larry Putnam and his associates at QSM (Quantitative Software Management). The main part of SLIM, the Software Equation, is shown in Table 2.1.

Effort Equation	Parameters
<p>Complete Model:</p> $Effort = 12^5 B \left(\frac{SLOC}{P} \right)^3 \frac{1}{Schedule^4}$ <p>Simplified Model:</p> $Effort = 56.4B \left(\frac{SLOC}{P} \right)^{\frac{9}{7}}$ $Effort = 8.14 \left(\frac{SLOC}{P} \right)^{\frac{3}{7}}$	<p>P: Productivity Parameter B: Special Skill Factor SLOC: Source Lines of Code</p>

Table 2.1. SLIM Software Equation

SLIM is different from other software estimation models discussed later in that the schedule equation needs to be solved before the estimation of an effort for a software product can be made. The Complete Model equation shown in Table 2.1 is derived from data of over 5000 projects collected by Larry Putnam and his associates at QSM. The Complete Model equation indicates that there is trade-off between effort and schedule. The software development effort gets lower as the software development schedule increases. The first equation of the Simplified Model is derived from projects that achieve their minimum schedule. It can be used for minimal or large projects where the effort is greater than or equal to 20 staff-months. The second equation of the Simplified Model can be used for the projects for which the computed minimum development schedule is greater than 6 calendar months.

The productivity parameter, P, is required to be determined prior to the effort estimation. For the determination of the productivity parameter, P, Putnam provides representative productivity parameters ranging from 754 to 3524578 for various types of systems by clustering around certain discrete numbers (Lawrence H. Putnam & Ware Myers, 1992). A simple scale of integer values is called the Productivity Index (PI) and is assigned to those clusters. If there is no similar type of system in the database, SLIM allows the user to determine the productivity parameter as a function of a project's use of modern programming practice, hardware constraints, personnel experience, interactive development, and other factors by using historical data within an organization (Boehm, B. W., 1984). However, it is difficult to produce accurate parameters for a given system's development.

The Productivity Index (PI) is a macro-level performance indicator which measures project environments, tools, staff, complexity of the product and so on. It has a scale ranging from 1 (low) to 36 (high). For example, if a low value is assigned for a project, its PI is associated with primitive environments, poor tools, unskilled and untrained people, weak leadership, ineffectual methods, or a high degree of complexity in the product. Since all environmental factors for the software product are aggregated into the productivity parameter, it is impossible to individually assess the impact of CASE tools on the software development productivity.

2.1.2 Jensen Model

The Jensen model is a software development schedule/effort estimation model which incorporates the effects of any of the environmental factors impacting the software development cost and schedule (Randall W. Jensen, 1984). This model is an empirical model that is similar to Putnam's SLIM. Jensen proposed his software equation which relates the effective size of the system and the technology to the implementation of the system. Jensen's software equation is formulated as

(EQ 1)

$$S_e = C_{te} \cdot \sqrt{E} \cdot t_d$$

$$E = 0.4 \cdot \left(\frac{S_e}{C_{te}}\right)^2 \cdot \frac{1}{t_d}$$

where

S_e = effective system size, t_d = development schedule

C_{te} = effective developer technology constant

E = lifecycle effort

The effective technology constant, C_{te} , measures the factors such as the availability of computing resources, organizational strategies, development tools and methodologies, personnel capability and experience, and familiarity with development and targeted computers. The effective technology constant contains two aspects of the production technology; technical and environmental. The technical aspects include the factors dealing with the basic development capability such as organization capability and experience, development practices, and tools. Those factors determine a basic technology constant ranging from 2000 and 20,000, with higher values characteristic of higher tool quality as shown in Figure 2.1.

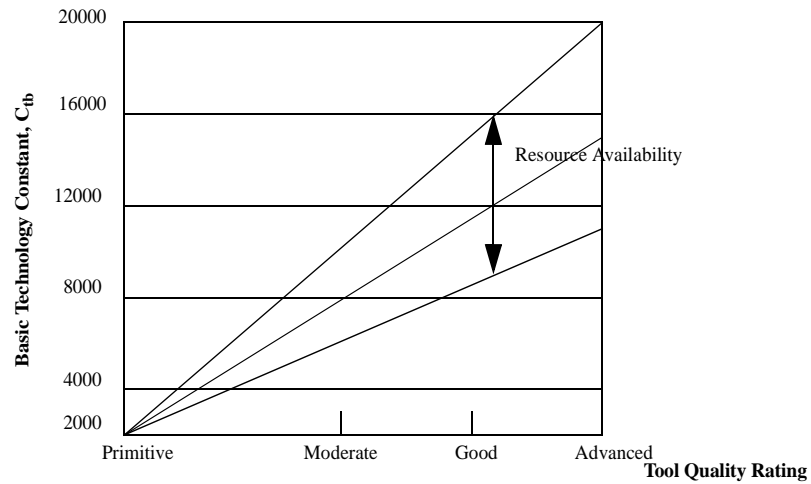


Figure 2.1. Basic Technology Constant

The environmental aspects are explained through the use of 13 adjustment factors in Table 2.2 applied to the basic technology constant as

(EQ 2)

$$C_{te} = C_{tb} \cdot \prod_{i=1}^{13} f_i$$

where

C_{tb} = basic technology constant
 $f_i = i_{th}$ environmental factor

The impact of the environmental factors and the basic technology constant on the development effort can be introduced by replacing the technology constant in (EQ 1) with (EQ 2). The effort ratio of the technology constant is summarized in Figure 2.2. In the case of Tool Quality, the effort ratio is between 1.36 to 1.51 influenced by some of the environmental adjustment factors which are similar to the COCOMO effort adjustment factors discussed later. The effort ratio indicates the impact of very poor tools on a

software project is a cost increase of 36 to 51 percent depending on the available resources over the cost achievable by an organization's set of the chosen tools.

Number	Environmental Factors
1	Special Display Requirement
2	Operational Requirement Detail and Stability
3	Real Time Operation
4	Memory Constraint
5	CPU Time Constraint
6	Virtual Machine Experience
7	Concurrent ADP Hardware Development
8	Developer Using Remote Computer
9	Development at Operational Site
10	Development Computer Different than Target Computer
11	Development at Multiple Sites
12	Programming Language Experience
13	System Reliability

Table 2.2. Constant Adjustment Factor

Jensen's model considers the quality of software tools as one of the factors which affects effort estimates. However, it does not capture what tasks are covered, what degree of tool integration is supported, how mature software tools are in the CASE market, and what kind of services are supported by software vendors. An updated version of the Jensen model known as SEER is provided and evolved by Galorath Associates.

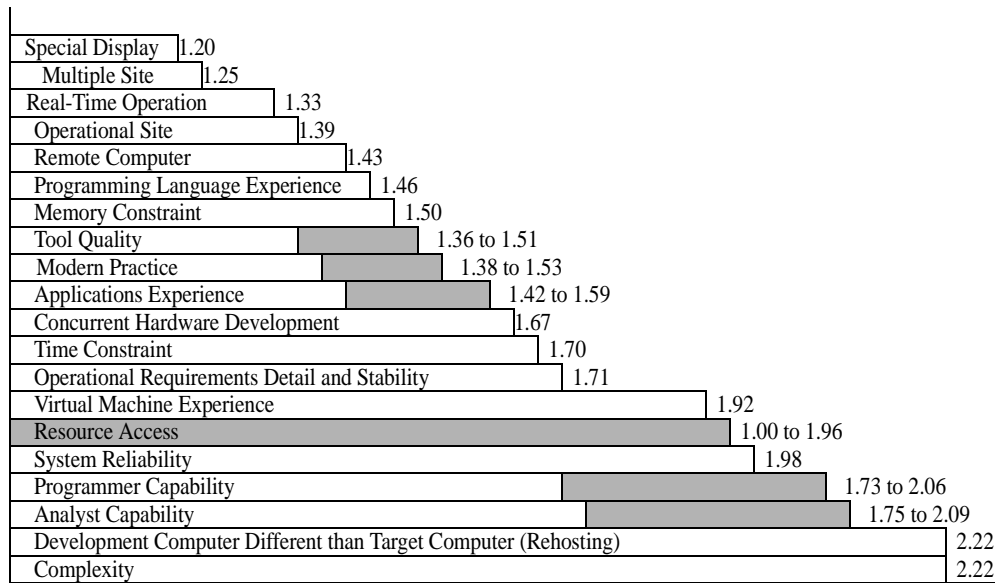


Figure 2.2. Jensen Software Development Effort Ratio

2.1.3 Checkpoint

Checkpoint is an automated, knowledge-based software estimation tool developed by SPR (Software Productivity Research) (Caper Jones, 1996). It uses the proprietary questionnaire designed by Capers Jones for carrying out large-scale baseline studies. It also uses the proprietary knowledge database of over 8000 software projects. The questionnaire consists of multiple choice questions built around a weight scale, the software effectiveness level (SEL). The SEL scale ranges from 1 to 5 with 3 representing the approximate U.S. average. A number lower than 3 represents situations that are better than the average, while a number higher than 3 represent situations that are worse and more hazardous than the average. The SEL scale is re-calibrated annually by the SPR based on their proprietary database. It uses the Function Point and Feature Point as its

sizing inputs. Its inputs focus on skill, experience, methods and tools, managerial and technical cohesiveness, and so on.

The inputs for assessing the productivity impact of CASE tools focus on the design automation environment, CASE Integration, project source code library, project document library, support software novelty, support software effectiveness, and programming debugging tools. Caper Jones suggests macro and micro estimation for the adjustment of the function points (Caper Jones, 1998). The macro estimation aggregates adjustment factors such as skill levels, tools, and methods into a single value. Table 2.3 shows an example of macro estimation by combining two factors, tools and team. In the macro estimation, it is difficult to assess the effect of individual factors on the software development productivity.

Factor Combinations	Multipliers
Superior team with superior tools	2.00
Superior team with average tools	1.75
Average team with superior tools	1.50
Superior team with marginal tools	1.25
Average team with average tools	1.00
Average team with marginal tools	0.90
Novice team with superior tools	0.75
Novice team with average tools	0.65
Novice team with marginal tools	0.50

Table 2.3. Example: Adjustment Assumption of Macro estimation

In the micro estimation, each factor is applied individually rather than concurrently with other factors for the adjustment. This micro estimation allows one to assess the

impact of each individual factor. Table 2.4 shows the multiplier values of team and tools for the function point adjustment. For example, if an average team with average tools has a productivity rate of 10 function points per staff month, any other combination can be explored like the following:

Excellent team with excellent tools:

$$10 * 1.5 * 1.2 = 18 \text{ function points per staff month}$$

Poor teams with poor tools:

$$10 * 0.5 * 0.8 = 4 \text{ function points per staff month}$$

Evaluation	Team capability multipliers	Tool capability multipliers
Excellent	1.50	1.20
Good	1.25	1.10
Average	1.00	1.00
Below average	0.75	0.90
Poor	0.50	0.80

Table 2.4. Example: Adjustment Assumptions for Microestimation

To determine the multiplier values for the “tool use” factor, Checkpoint requires the input of tool effectiveness ranging from 1 (Excellent) to 5 (Poor). Tools are categorized by the phases (Project Management, Requirements, Design, Coding, Testing and Debugging, Documentation, and Maintenance methods/tools) they support. Checkpoint captures the information about the degree of tools integration through the input of CASE Integration. The criteria for the CASE Integration is shown in Table 2.5. Even if Checkpoint evaluates

CASE tools by quality and the degree of integration across phases/projects, it does not account for the effect of tool maturity and user support.

Poor	Below Average	Average	Good	Excellent
No use of CASE tools in the company/organization	Company/organization is using CASE tools, but not on this project	Limited amount of CASE on this project	Some use of CASE tools in multiple phases	Integration of CASE across all phases of lifecycle

Table 2.5. CASE Integration Criteria

2.1.4 PRICE-S

PRICE-S is an empirical, parametric, shared, and macro software estimation model originally developed by Martin Maritta Price Systems (initially RCA Price, GE Price, then Price Systems) as a family of models for hardware and software cost estimation developed in 1977 by F. Freiman and R. Park (Robert E. Park, 1988; Parametric, 1995). Even though PRICE-S is developed as a proprietary model for the estimation of military software projects, it became popular model and is now available by PRICE Systems. The current PRICE-S model consists of three sub models: the Acquisition Model, the Sizing Model, and the Life-cycle Cost Model. The detailed description of those models are available in the Price Systems' web page (PRICE, 1998). The PRICE-S model is currently adopted by ForeSight to address non-military software projects only. ForeSight has been tuned to the commercial and non-military government records for the PRICE-S database.

PRICE-S performs a natural sequence of explaining the core computations to transform product descriptions into project estimates. Table 2.6 describes the flow of the

sequence briefly. Detailed information on the core computation is available in (Robert E. Park, 1988).

No	Name	Description
1	Build Composite Descriptor	The application mix and fractions of new design and new code for eight functional categories are used to construct weighted averages, which serve as composite indices to describe the characteristic of the application and the effective amount of new design and new code.
2	Size the Job	The magnitude of each of the three major development activities (design, implementation, integration & testing) are estimated relative to the efforts that would take place were there no reusable design and reused code are not free, and that reuse has different consequences for each phase of software development.
3	Estimate a Normal Schedule	Effective masses for major activity are derived as functions of software volume (size), density (application) and component reuse (new design and new code). The concept of weight is introduced and used as a measure for software mass.
4	Adjust Software Weights for Project Size	The normal schedule is used to adjust the software weights for the three core phases. This adjustment accounts for the effects that project size has on application difficulty.
5	Estimate Key Costs	Key costs are estimated for each core development phase. PRICE S views systems engineering during design as the key to software development.
6	Estimate Phases and Element Costs	The core process breaks each phase down into five categories of effort such as systems engineering and design, programming, configuration management and quality assurance, documentation, and program management. Estimates for these activities are obtained from a flow-down model of ripple effects.
7	Adjust Costs to Match Local Experience	The cost elements from stage 6 are multiplied by corresponding factors from the estimator's customized cost adjustment table.
8	Satisfy Constraints on Time and Resources	Schedules and costs are adjusted to account for the effects of mandated milestones and limited resources.
9	Inflate to Obtain As-Spent costs	If as-spent costs are requested, a profiling submodel is applied to each major activity to distribute the effects of inflation over time.
10	Convert Costs to Output Format	If requested, markups for G & A and profit are applied, and costs are converted from dollars to specified manpower or currency units (person-hours, person-months, pounds, lira, marks, etc.). the results are scaled, per instruction, and a report is prepared.

Table 2.6. PRICE-S Core Computation Flow

In stage 3, PRICE-S uses complexity adjustment factors such as Personnel, Productivity familiarity, Software tools, and Complicating factors to estimate a nominal schedule. The model is formulated as

(EQ 3)

$$T_D = 0.027 \cdot F \cdot Wt_D^{0.37} \cdot CPLX^{1.3}$$

$$X_C = 0.013 \cdot F \cdot Wt_C^{0.37} \cdot CPLX^{0.95}$$

$$X_T = 0.024 \cdot F \cdot Wt_T^{0.37} \cdot CPLX$$

Where

T_D : Time to perform design

X_C : Extension to complete coding and unit testing

X_T : Extension to complete integration and testing

Wt_D : Adjusted weight of software for design

Wt_C : Adjusted weight for coding and unit testing

Wt_T : Adjusted weight for integration and testing

F : Parametric function that accounts for other factors such as the developer's resources, the nature of the customer and requirements, and the state of art in software development

The complexity factors account for a fundamental difference between the inherent complexity of a task and the apparent complexity of the product of the task. The complexity value is determined as

(EQ 4)

$$CPLX = (\text{Reference Value}) + (\text{Complications})$$

The normalized value of 1.0 is used for the default Reference Value and then the selected values of the adjustment factors shown in Table 2.7 are added to get the adjusted project complexity value.

PERSONNEL	SOFTWARE TOOLS
Outstanding crew.....-0.2	Very highly automated.....-0.2
Extensive experience.....-0.1	Highly automated.....-0.1
Normal crew.....0	Nominal.....0
Mixed experience.....+0.1	Low.....+0.1
Relatively inexperienced.....+0.2	Very Low.....+0.2
PRODUCT FAMILIARITY	COMPLICATING FACTORS
Familiar type of product.....-0.1	New Language.....+0.2
Normal, new product.....0.0	Changing requirements.....+0.2
New Line of business.....+0.2	

Table 2.7. Complexity Adjustment Factors

PRICE-S also uses “software tools” as one of the environmental factors to estimate the normal schedule. Since its value is aggregated into the complexity value, it is difficult to individually evaluate the impact of CASE tools on the software development productivity.

2.1.5 Softcost

Softcost is a mathematical resource and schedule estimation model developed by Dr. Robert Tausworthe (1981) of the NASA Jet Propulsion Laboratory. This model borrows the good features from a number of different software estimation models such as PRICE-S, SLIM, Waltson-Felix and so on. Softcost uses 68 parameters which relate to productivity, duration, staffing level, documentation, and computer resources. It uses another 69 parameters to apply the estimated effort, staff, and duration to a Work Breakdown Structure (WBS). It applies a modification of Putnam’s SLIM to check on the

feasibility of resource estimates. The main outputs of the model are effort, schedule, staff productivity, computer resources, staffing, and documentation.

SoftCost-R was developed to be applicable to all types of projects as a modification of the Softcost model (D. Reifer, P.Kane, & D.Willens, 1989). As a part of family of model, SoftCost-Ada was developed to match the object-oriented and reuse paradigm (D. Reifer, P.Kane, & T. Pillsbury, 1989). A key input of SoftCost-R is size, either in SLOC or computed SLOC from function points. It also provides a more sophisticated sizing submodel. The SoftCost-R uses four categories (product, computer, personnel, and project) of parameters like COCOMO. However, it uses more than thirty unique parameters such as peer review, customer experience, and degree of standardization. The mathematical representation of the effort equation in the model is shown as the following.

(EQ 5)

$$Effort = GAMMA \cdot P_0 \cdot A_1 \cdot A_2 \cdot (Size)^B$$

Where

GAMMA: Application domain specific effort calibration

P₀: Constant factor calculated from calibrated values

A₁: Reifer cost drivers A₂: Productivity adjustment factors

B: Calibrated effort exponent

SoftCost-R uses two tool-related cost drivers, *Use of Software Tools/Environment* and *Software Tool/Environment Stability* which are parts of A₂ in (EQ 5). RCI (Reifer Consultants, Inc.) has found that these factors are highly correlated. The SoftCost-R accounts for this correlation using the matrix in Table 2.8 for the adjustments of estimates. SoftCost-R captures the productivity impact of software tools/environment on effort

through those factors. However, it does not take into account the maturity of tools and user support from software vendors.

Software Tool/ Environment Stability	Use of Software Tools/Environment					
	Very Low	Low	Nominal	High	Very High	Extra High
Very Low	1.19	1.16	1.12	1.08	1.00	1.00
Low	1.13	1.10	1.07	1.00	0.94	0.90
Nominal	1.90	1.06	1.00	0.95	0.91	0.87
High	1.50	1.00	0.96	0.92	0.89	0.85
Very High	1.00	0.96	0.91	0.89	0.85	0.79

Table 2.8. Use of Software Tools and Software Tool/Environment Stability Matrix

2.1.6 COCOMO

COCOMO (Constructive Cost Model) is a software estimation package of three models (Basic, Intermediate, and Detailed) proposed by Barry Boehm, B. W. (Boehm, B. W., 1981). The Basic COCOMO provides quick, early, rough-order of magnitude effort and schedule estimates in the early stage of the project development without considering the project factors that account for the differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and other attributes known to have a significant influence on software costs. The Basic COCOMO effort equation is a non-linear function of the project size shown in (EQ 6). A (Coefficient) and B (Exponent) take different values according to the development modes such as organic, semi-detached, and embedded which explains the differences in organizational understanding of the project, experience in working with related software system, project size range and so on.

(EQ 6)

$$Effort = A(Size)^B$$

The Intermediate COCOMO is a compatible extension of the Basic COCOMO. It uses an additional 15 predictor variables called Cost Drivers or Effort Multipliers in Table 2.9 which have an influence on software development effort and schedule. The cost drivers are multiplicatives in (EQ 7) to adjust a nominal estimate that does not reflect the project attributes.

(EQ 7)

$$Effort = A(Size)^B \prod_{i=1}^{15} EM_i$$

The Detailed COCOMO accounts for the impacts of the cost drivers on the efforts from different in the development lifecycle. It provides phase-sensitive effort multipliers used to determine the required effort for each phase and a three-level product hierarchy (*module, subsystem, and system levels*) to avoid tedious and repetitive inputs of cost drivers for different product components in the Intermediate COCOMO.

Product: Required Software Reliability (RELY) Database Size (DATA) Product Complexity (CPLX)	Platform: Execution Time Constraint (TIME) Main Storage Constraint (STOR) Virtual Machine Volatility (VIRT) Computer Turnaround Time (TURN)
--	--

Table 2.9. COCOMO Cost Drivers

Personnel: Analyst Capability (ACAP) Application Experience (APEX) Programmer Capability (PCAP) Virtual Machine Experience (VEXP) Programming Language Experience (LEXP)	Project: Use of Modern Programming Practice (MODP) Use of Software Tools (TOOL) Required Development Schedule (SCED)
--	--

Table 2.9. COCOMO Cost Drivers (Continued)

The Intermediate COCOMO and the Detailed COCOMO have a cost driver called TOOL to assess the impact of CASE tools on software productivity. The TOOL rating scale provides a categorization of the typical set of tools available at each level: Very Low, Low, Nominal, High, and Very High. However, the rating scale requires some kind of judgement to rate any particular set of tools and the quality/degree of the tool environment.

Ada COCOMO is an updated version of COCOMO to determine the effect of Ada and the Ada process model on software development efforts and schedules (Boehm, B. W. & Walker Royce, 1989). The effort equation of the model is shown in (EQ 8). Unlike the COCOMO, it uses the Ada process model factor for the scaling effect on project effort rather than a fixed exponent value. The Ada process model factor has four elements: experience with the Ada process model; design thoroughness: unit package specs compiled, bodies outlined; risks eliminated by PDR; and requirements volatility during development. Each element has a scale from 0.00 to 0.05. The Ada COCOMO uses 18 cost drivers rather than 15 cost drivers used in COCOMO. It adds two new cost drivers, Required Reusability (RUSE) and Classified Security Application (SECU). The Virtual

Volatility (VIRT) of COCOMO is splitted into host volatility (VMVH) and target volatility (VMVT).

(EQ 8)

$$Effort = 2.8(Size)^{1.04 + \sum_{i=1}^{18} w_i} \cdot \prod_{i=1}^{18} EM_i$$

The TOOL cost driver of COCOMO is extended with two additional rating levels: Extra High and XX-High to reflect more fully populated and integrated tool sets. Table 2.10 shows the differences between the TOOL rating values in COCOMO and Ada COCOMO.

Rating Level	Rating Scale	COCOMO	Ada COCOMO
Very Low	Very few, primitive tools	1.24	1.24
Low	Basic micro tools	1.10	1.10
Nominal	Basic mini tools	1.00	1.00
High	Basic maxi tools	0.91	0.91
Very High	Extensive tools, little integration	0.83	0.83
Extra High	Moderately integrated environment (UNIX)		0.73
XX High	Fully integrated environment		0.62
Productivity Range		1.49	2.00

Table 2.10. TOOL Rating in COCOMO and Ada COCOMO

COCOMO has been updated to COCOMO II since 1994 to address the issues on non-sequential and rapid development process, reengineering, reuse-driven approach, and so on. The COCOMO II has three submodels: Application Composition, Early Design,

and Post Architecture, to deal with the current and future software marketplace. COCOMO II will be discussed later in more detail.

2.1.7 Overall Summary of Cost Estimation Models

Most of the software estimation models described in previous sections consider the effect of CASE tools as one of the project factors to predict an adjusted project effort. Since some models are proprietary, it is difficult to know the application of the tool-related factors in the assessment of the impact and the productivity ranges in their model. Even though many studies and much evaluation criteria have emphasized the importance of the tool maturity and user support, all of the software estimation models do not capture CASE tool impacts of those factors on the software development effort to predict an accurate effort estimate. Table 2.11 summarizes the productivity dimensions the software estimation models support on the basis of CASE tools.

While SLIM and COCOMO do capture the influence of CASE tools' activity coverages and integration on estimating project efforts as shown in the Table 2.11, the rest of the models capture the impact of CASE tools on the basis of only Completeness of Activity Coverage. As mentioned earlier, their productivity dimensions are aggregated into one of the environmental factors. Thereby, it is difficult to assess the individual productivity impact.

	SLIM	Jensen	Checkpoint	PRICE-S	Softcost	COCOMO
“use of software tools”	Yes	Yes	Yes	Yes	Yes	Yes
Completeness of Activity Coverage	No	No	Yes	No	Yes	Yes
Degree of Tool Integration	No	No	Yes	No	No	Yes
Tool Maturity and User Support	No	No	No	No	No	No

Table 2.11. Summary of Cost Estimation Models

2.2 Tool Impact on Software Productivity

Besides the software cost estimation models mentioned earlier, there has been a lot of research efforts to assess the impact of CASE tools on software development productivity. The two survey result shown in Figure 2.3 conducted on users of the ITE (Index Technology’s Exclerator) product shows that users gain an average of 30- to 40-percent productivity improvement in the analysis/design phases and 11- to 14-percent improvement in the maintenance phase by using ITE product environments (Elliot J. Chikofsky & Burt L. Rubenstein, 1988). But many of the user organizations report that their key goal of using CASE tools is in quality improvement of the developed systems. They can find errors/ inconsistencies and refine specifications easily in the earlier stages of the production life cycle. This leads to the productivity improvement by automated error and inconsistency checking.

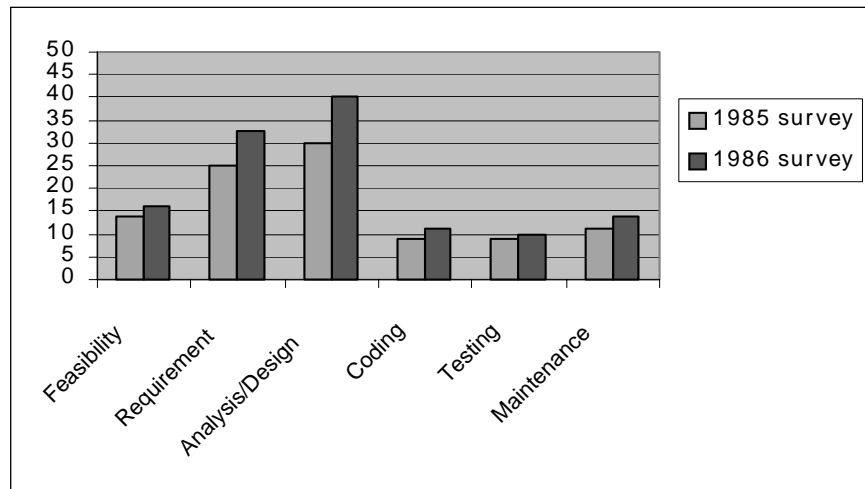


Figure 2.3. Productivity Improvement in a lifecycle phases

The survey of users on 22 different projects conducted at 14 companies in Europe analyzed the degree of facilitating project tasks supported by the CASE environment EPOS. All degrees in the lifecycle phases have positive values from 0.0 to 1.25 on a scale of -2 (much harder) to +2 (much easier) indicating higher productivity as shown in Figure 2.4. It shows the average degree for a whole lifecycle phases has a high value of 0.74 (Peter Lempp & Rudolf Lauber, 1988). The use of CASE tools for project management and control has the highest relative productivity gain throughout the whole lifecycle.

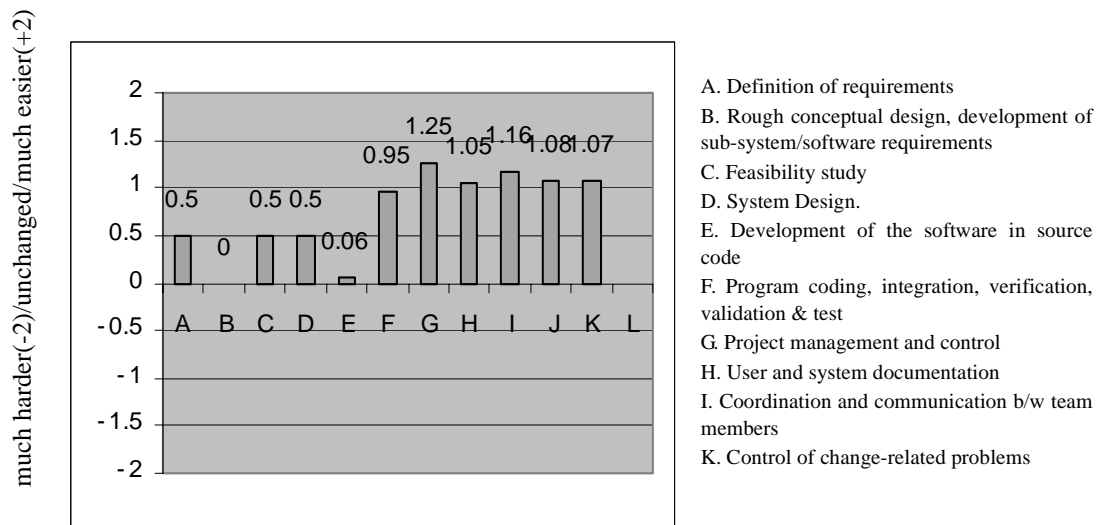


Figure 2.4. Degree of facilitating project tasks (indicating higher productivity)

In (Caper Jones, 1998), the maximum positive and negative productivity impacts of technology adjustment factors which affect predicting project effort are presented. Table 2.12 shows a part of the result focusing on CASE tool productivity impact. It indicates that negative results are much larger than those of the positive results for technical CASE tools. That means it is easier to make mistakes and lower the productivity when an inappropriate tool set is used.

Factor	Positive, Plus Range (%)	Negative, Minus Range (%)
Management tools	30	-45
Technical CASE tools	27	-75
Quality estimating tools	19	-40

Table 2.12. Impact of Software Tools on Productivity

3.0 COCOMO II and CASE Classification

3.1 COCOMO II and Its Database

The COCOMO II is one of the most popular and widely-adopted software cost estimation models. It has been updated from the original COCOMO (Boehm, B. W., 1981) and its Ada successor (Boehm, B. W. & Walker Royce, 1989) in order to address the issues on new process models and capabilities such as concurrent, iterative, and cyclic processes; rapid application development, COTS (Commercial-Off-The-Shelf) integration and software maturity initiatives. The COCOMO II research effort targets the estimation of software projects in the 1990's and 2000's. The definition and rationale are described in (Boehm, B. W. et al., 2000). The initial definition of the model has been refined as actual project data points are collected and analyzed.

The COCOMO II provides a tailorable mix of three submodels: *Application Composition Model*, *Early Design Model*, and *Post-Architecture Model*. These are consistent with the granularity of the available information in a software development lifecycle in order to support software estimation. The Post-Architecture model estimates the effort and the schedule involved in the development and maintenance of a software product after the lifecycle architecture of the software product is established. It is the most mature model among the three models and has been calibrated for more accurate estimation. This chapter discusses the Post-Architecture Model and the analyzed result of the COCOMO II database focusing on the TOOL effort multiplier. It also describes a

classification of available CASE tools and technologies based on the breadth of coverage in a software development lifecycle.

3.1.1 COCOMO II Post Architecture Model

The COCOMO II Post-Architecture model is the most detailed model and is applied when the system architecture is developed. It can be used in sectors of the future software marketplace shown in Figure 3.1 such as Application Generators, System Integration or Infrastructure to provide the estimates of effort and schedule for the actual development and maintenance of a software project (Boehm, B. W. et al., 2000). This model has about the same granularity as the intermediate model of the original COCOMO and Ada COCOMO models.

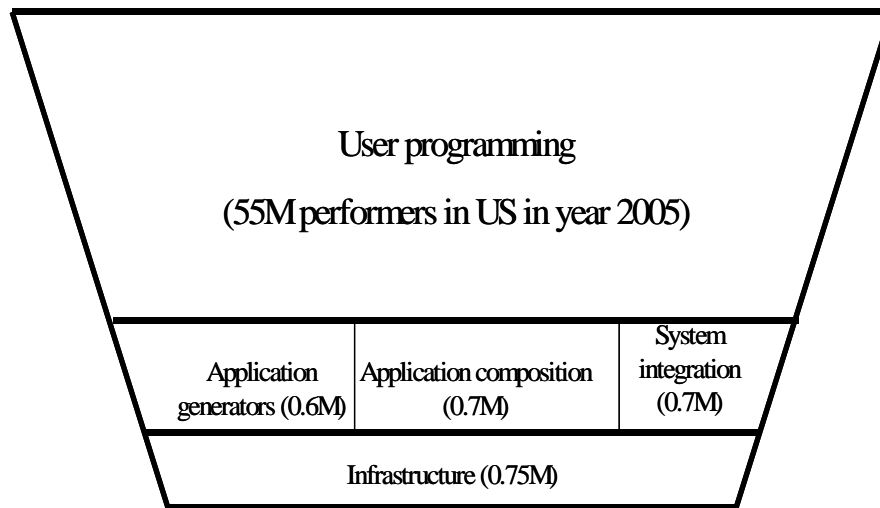


Figure 3.1. The Future of Software Market Place

(EQ 9) is the base model for COCOMO II Early Design and Post-Architecture models. Many of software estimation models mentioned in Chapter 2 use this equation to support linear and non-linear effects of the project factors. The effort estimate of the COCOMO II is expressed in terms of a Person Month (PM), which is the amount of time one person spends for the project in one calendar month. The key inputs for the base model are the Size of the developing software project, a constant A, and a scale factor B. The COCOMO II uses Object Points (Banker R., Robert J. Kauffman, & Rachna Kumar, 1992), Source Lines of Code (Robert E. Park, 1992), and Unadjusted Function Points (IFPUG, 1994) with modifiers for reuse, adaptation, and software breakage as sizing methods. The multiplicative constant, A, captures the linear effects of effort as the project size increases. It is also used to calibrate the model for a better fit.

(EQ 9)

$$PM_{nominal} = A \times (Size)^B$$

The B in (EQ 9) is the exponential factor that accounts for the relative economies or diseconomies of scale in different size of software projects. For example, if $B < 1.0$ and the project size is doubled, then the effort for the project is less than doubled.

<p>If $B < 1.0$: Economies of scale. If $B = 1.0$: Economies and Diseconomies in balance. If $B > 1.0$: Diseconomies of scale.</p>

The three modes of the original COCOMO (Organic, Semi-detached, and Embedded) respectively have different exponents: 1.05, 1.12, and 1.20. The scaling

models in those three modes exhibit the diseconomies of scale. The exponent B of Ada COCOMO varies from 1.04 to 1.24, depending on the project's progress in reducing diseconomies of scale via early risk elimination, solid architecture, stable requirements, and Ada process maturity. It also displays the diseconomies of scale as the project size increases. The detailed information about the economies and the diseconomies of scale is available in (Banker R., Chang, & C. Kemerer, 1994).

The COCOMO II uses five scale factors such as precedentedness, development flexibility, architecture/risk resolution, team cohesion, and process maturity that combines the original COCOMO and Ada COCOMO scaling approaches into a single rating-driven model. The scale exponent B is determined via the formula shown in (EQ 10).

(EQ 10)

$$B = 0.91 + 0.01 \times \sum_{i=1}^5 SF_i$$

The list of those scale factors is shown in Table 3.1. The Precedentedness (PREC) and the Development Flexibility (FLEX) capture the differences in the three modes (Organic, Semi-detached, and Embedded) of the original COCOMO (Boehm, B. W., 1981). The Architecture/Risk Resolution (RESL) is the scale factor that combines Design Thoroughness and Risk Elimination by Product Design Review (PDR) in the Ada COCOMO (Boehm, B. W. & Walker Royce, 1989). The Team Cohesion (TEAM) captures the difficulties in the cooperation of the project stakeholders such as user, customer, developer, etc. The Process Maturity (PMAT) replaces the Ada COCOMO process

maturity factor with the Software Engineering Institute's CMM (Capability Maturity Model) level (Paulk, M. C. et al., 1993). There are two ways to determine the rating of the Process Maturity. The first one is based on overall maturity level. The second one is based on 18 Key Process Areas (KPAs) in the SEI CMM. The detailed procedure to determine the rating of the PMAT is described in (Boehm, B. W. et al., 2000).

Sym .	Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
SF ₁	Precedentedness (PREC)	thoroughly unprecedented	largely unprecedented	somewhat predated	generally familiar	largely familiar	thoroughly familiar
SF ₂	Development Flexibility (FLEX)	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF ₃	Architecture/Risk Resolution (RESL)	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF ₄	Team Cohesion (TEAM)	very difficult interaction	some difficult interaction	basically cooperative interaction	largely cooperative	highly cooperative	seamless interactions
SF ₅	Process Maturity (PMAT)	CMM 1 (Lower)	CMM 1 (Upper)	CMM 2	CMM 3	CMM 4	CMM 5
		or an average of KPA Goal Compliance					

Table 3.1. Scale Factor Ratings of the COCOMO II

The COCOMO II uses the Effort Multiplier (EM), (or Cost Drivers) to capture the characteristics of the developing software project to adjust the nominal effort estimate predicted via the formula in (EQ 11). The adjusted effort formula for the COCOMO II Post Architecture model is as follows:

$$PM_{adjusted} = PM_{nominal} \times \prod_{i=1}^{17} EM_i$$

The Post Architecture has 17 Effort Multipliers that have a multiplicative effect on the effort estimation, while Early Design Model has 7 Effort Multipliers. Those Effort Multipliers are grouped into 4 categories: *Product*, *Platform*, *Personnel*, and *Project*. Table 3.2 shows the list of Effort Multipliers and a short description of those multipliers, which take advantage of the greater knowledge available in the development stage of a software lifecycle.

Category	Sym.	Effort Multiplier	Description
Product	EM ₁	Required Software Reliability (RELY)	This is the measure of the extent to which the software must perform its intended function over a period time. If the effect of a software failure is only slight inconvenient, then RELY is low. If a failure would risk human life, then RELY is very high.
	EM ₂	Database Size (DATA)	This measures attempts to capture the effect large data requirements have on product development. The reason the size of the database is important to consider it because of effort required to generate the test data that will be used to exercise the program.
	EM ₃	Product Complexity (CPLX)	This 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.
	EM ₄	Required Reusability (RUSE)	This accounts for the additional effort to construct components intended for reuse on the current or future projects. This effort is consumed with creating more generic design of software, more elaborate documentation, and more extensive testing to ensure components are ready for use in other applications.
	EM ₅	Documentation match to life-cycle needs (DOCU)	The rating for this cost driver is evaluated in terms of the suitability of the project's documentation to its life-cycle needs. The rating scale goes from Very Low (many lifecycle needs uncovered) to Very high (very excessive for life-cycle needs)
Platform	EM ₆	Execution Time Constraint (TIME)	This is a measure of the execution time constraint imposed upon a software system. The rating ranges from nominal (less than 50%) of execution time resource used, to extra high (95%) of execution time resource is consumed.
	EM ₇	Main Storage Constraint (STOR)	This rating represents the degree of the main storage constraint imposed on the software system or subsystem. the rating ranges from nominal (less than 50%) to extra high (95%)
	EM ₈	Platform Volatility (PVOL)	“Platform” is used to mean the complex of hardware and software (OS, DBMS, etc.) the software product calls on to platform its tasks. The platform includes any compilers or assemblers supporting the development of the software system. This rating ranges from low, where there is a major change every 12 months, to very high, where there is major change every two weeks.

Table 3.2. Post Architecture Effort Multipliers

Category	Sym.	Effort Multiplier	Description
Personnel	EM ₉	Analyst Capability (ACAP)	Analysts are the personnel that work on the requirements, high level design and detailed design. The major attributes that should be considered in this rating are Analysis and Design ability, efficiency and thoroughness, and the ability to communicate and cooperate. The rating should not consider the level of experience of the analyst; that is rated with APEX. Analysts that fall in the 15th percentile are rated very low and those that fall in the 95th percentile are rated as very high.
	EM ₁₀	Programmer Capability (PCAP)	The evaluation of this cost driver 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. The experience of the programmer should not be considered here; it is rated with APEX. A very low rated programmer team is in the 15th percentile and a very high rated programmer team is in the 95th percentile.
	EM ₁₁	Application Experience (APEX)	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. A very low rating is for application experience of less than 2 months. A very high rating is for experience of 6 years or more.
	EM ₁₂	Platform Experience (PLEX)	This cost driver captures the importance of understanding the use of more powerful platforms, including graphic user interfaces, database, networking, and distributed middleware capabilities.
	EM ₁₃	Language and Tool Experience (LTEX)	This is a measure of the level of programming languages and software tool experience of the project team developing the software system or subsystem. A low rating is given for experience of less than 2 months. A very high rating is given for experience of 6 or more years.
	EM ₁₄	Personnel Continuity (PCON)	The rating scale for PCON is in terms of the project's annual personnel turnover: from 3%, (very high), to 48%, (very low).

Table 3.2. Post Architecture Effort Multipliers (Continued)

Category	Sym.	Effort Multiplier	Description
Project	EM ₁₅	Use of Software Tools (TOOL)	The tool rating ranges from simple edit and code, (very low), to integrated lifecycle management tools, (very high).
	EM ₁₆	Multisite Development (SITE)	Determining this 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 fully interactive multimedia)
	EM ₁₇	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. Accelerated schedules tend to produce more effort in the later phases of development because more issues are left to be determined due to lack of time to resolve them earlier. A schedule compression of 74% is rated very low. A stretch-out of a schedule produces more effort in the earlier phases of development where there is more time for thorough planning, specification and validation. A stretch-out of 160% is rated very high.

Table 3.2. Post Architecture Effort Multipliers (Continued)

Each Effort Multiplier for the Post-Architecture Model has a set of six qualitative rating levels from Very Low to Very High. The nominal rating for an Effort Multiplier is set to 1.0. Off-nominal values generally change the estimated effort. For example, if an EM weight is less than 1.0, it causes a reduction in the development effort for the project. The production of all 17 Effort Multiplier rating values is multiplied by the nominal effort estimate as shown in (EQ 11) in order to produce an adjusted effort estimate. Since this research is based on the COCOMO II Post-Architecture model, the 5 scale factors and 17 Effort Multipliers are used as the predictors of a statistical research model except that the extended new three dimensional tool rating scales discussed later are used instead of the COCOMO II one dimensional TOOL rating scale.

3.1.2 Analysis of the COCOMO II Database

The COCOMO II Post-Architecture model has been calibrated using actual project data collected from the affiliates of the Center for Software Engineering at the University of Southern California (USC-CSE) like other parametric software estimation models mentioned earlier. Those affiliates represent Commercial, Aerospace, Government, and FFRDC (Federally Funded Research and Development Centers) organizations. The initial calibration for COCOMO II.1997 was performed with 83 actual projects by using a 10% weighted multiple regression approach, which used 10% of the data-driven and 90% of the expert-judged values for the cost drivers (Bradford Clark, Sunita Chulani, & Boehm, B. W., 1998). Since the initial calibration, the size of the COCOMO II database has grown from 83 project data to 161 project data. The USC-CSE performed another round of calibration known as COCOMO II.1998 with the 161 project data by using a more sophisticated Bayesian approach to alleviate the problems caused by imprecise sizing, effort, and cost driver inputs in the 10% weighted multiple regression approach. The more detailed information on the COCOMO II Bayesian analysis is described in (Sunita Chulani, Boehm, B. W., & Bert Steece, 1999).

The data in the COCOMO II database is recorded from a data collection form asking questions regarding size, effort, schedule, 5 scale factors, 17 effort multipliers and quality information of the completed projects (USC-CSE, 1999). For the size input to the Post-Architecture model, Source Lines Of Code (SLOC) (Robert E. Park, 1992) or Unadjusted Function Points (UFP) (IFPUG, 1994) can be used as a counting method. Also, the size

input can be adjusted for breakage effects, adaptation, and reuse with the modifiers: the percentage of design modified (DM); the percentage of code modified (CM); the percentage of modification to the original integration effort required for integrating the reused software (IM); software understanding (SU); unfamiliarity with the software (UNFM); and assessment and assimilation (AA). The unit for the Software size is thousands of source lines of code (KSLOC). A frequently asked question is what defines source lines of code or Unadjusted Function Points. Even if the COCOMO II Model (Boehm, B. W. et al., 2000) defines logical lines of code, the collected data exhibits local variations in the interpretation of the counting rules. For the effort input, the COCOMO II uses a Person Month, which is 152 hours a month as the input unit. Since different organizations have different definitions of PM, this information is transformed using PM(152 hrs/month). Figure 3.2 shows the distributions of software size and actual effort of 161 projects in the COCOMO II database.

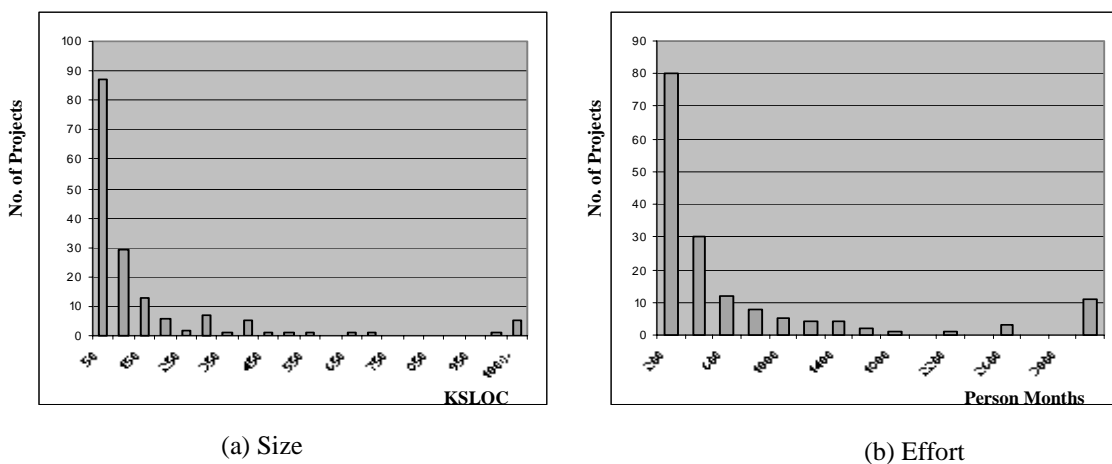


Figure 3.2. Distribution of Size and Actual Effort for the COCOMO II.1998 161 dataset

The histograms in Figure 3.2 indicate that the distributions of Size and Effort are very positively skewed to the right. This asymmetry is caused because the sample means of Size and Effort are much larger than the medians of Size and Effort as shown in Table 3.3.

	Mean	Median
Size	130.87	46.92
Effort	711.20	195.00

Table 3.3. Mean and Median Values for Size and Effort

For the scale factors and Effort Multipliers, the COCOMO II uses a rating scale ranging from Very Low to Extra High. Figure 3.3 shows the distribution of the TOOL rating scale that captures the productivity differences of CASE tools. This histogram is also skewed to the right (i.e. the dispersion is weak at the high end).

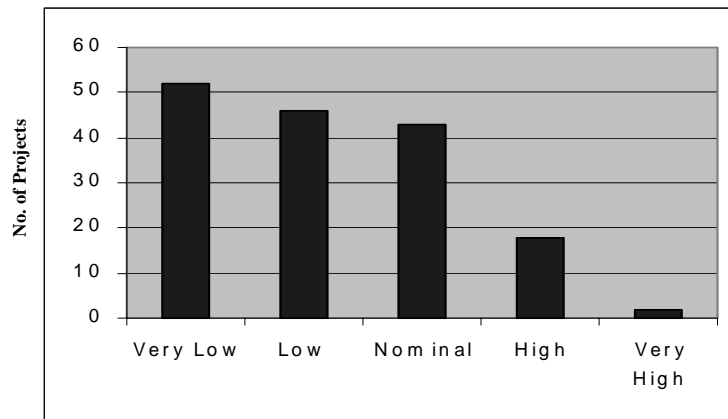


Figure 3.3. Distribution of the COCOMO II TOOL Cost Driver

The Bayesian Approach for COCOMO II.1998 Bayesian calibration (Sunita Chulani, Boehm, B. W., & Bert Steece, 1999) uses the a-priori (expert-judgment) information and sample (data-driven) information to determine the posteriori distribution for the model parameters. The weighted values for the two sources of information are assigned according to their variance (or precision). That is, if the variance of a-priori information is larger than that of sample information, a higher weight is assigned to the sample information. Figure 3.4 shows how the posteriori productivity range (ratio between the least productive parameter rating and the most productive parameter rating) for the TOOL (use of software tools) rating is determined using the two sources of information. In this case, a higher weight is assigned to the lower-variance sample information causing the posteriori mean to be closer to the sample mean. The t-value 2.4888 in Figure 3.4 is the ratio between the estimate and corresponding standard error in the log-transformed regression analysis for 161 project data. It shows that the predictor TOOL has statistically a strong significance (t-value > 1.96) in estimating the effort of software development. The practical interpretation of this result is that “increasing tool coverage makes a statistically significant difference in reducing software development effort, even after normalizing the effects of other variables such as process maturity. Given that previous anecdotal data still generates controversy on this point (Terry Bollinger, 2000; Bill Curtis, 2000), this is a valuable result.

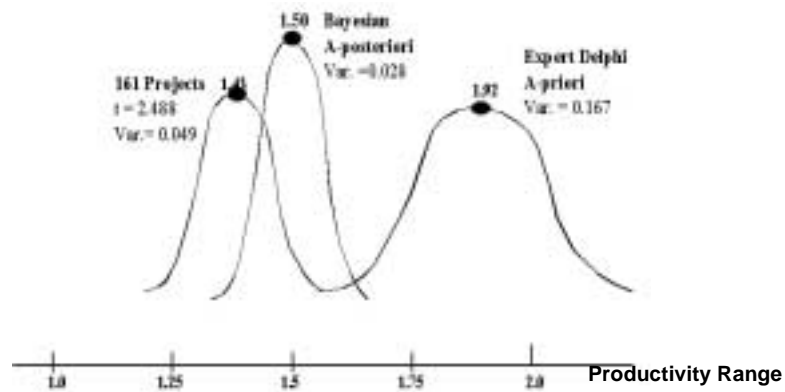


Figure 3.4. A-Posteriori Update for TOOL Rating Productivity Range

The productivity ranges for all parameters used in the COCOMO II Bayesian calibration is illustrated in Figure 3.5. This Figure delivers an overall perspective on the relative software productivity ranges. The productivity range for the TOOL parameter shows a relatively high-payoff in a software productivity improvement activity.

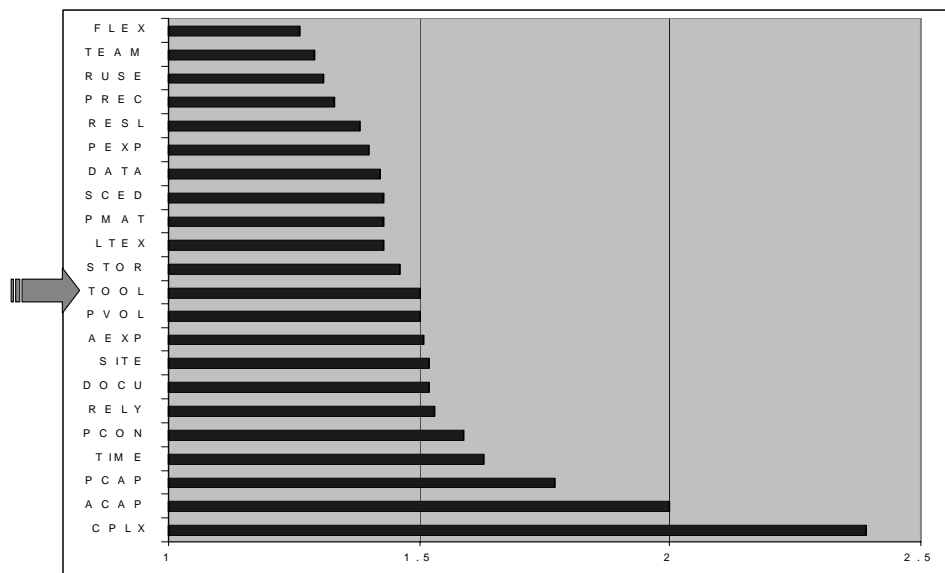


Figure 3.5. COCOMOII.1998 Productivity Ranges

As mentioned earlier, the COCOMO II Bayesian approach uses a multiple regression method to derive sample information from the COCOMO II database. The multiple regression method will be discussed later in detail. In the COCOMO II model, PM (Person-Month) is a response variable (or dependent variable). 5 Scale Factors, 17 Cost Drivers, and Project Size are predictor variables (or independent variables). The definition of those variables goes beyond this research. If a predictor variable in the model is not independent of other independent variables, it is very difficult to determine the contribution of the independent variable to software productivity activities. Figure 3.6 shows the correlations of other cost drivers, Size, and PM with the TOOL rating scale. All correlations of the variables with the TOOL rating are smaller than 0.45. That means the TOOL rating is largely orthogonal to other variables in the model.

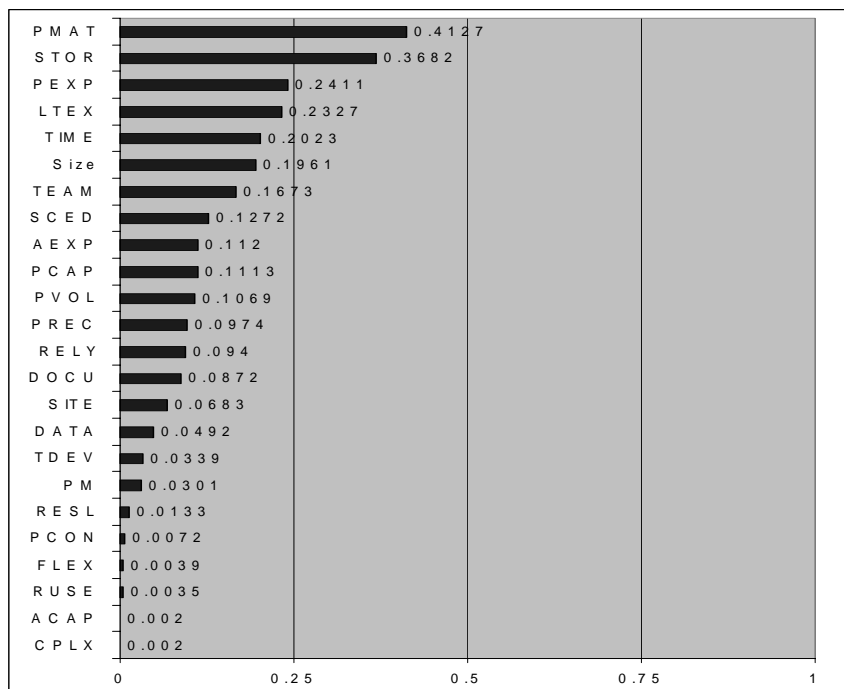


Figure 3.6. Magnitude of Correlations with TOOL

3.2 Software Tools Classification by Lifecycle Activity

The proliferation of CASE tools has compelled researchers in the CASE field to focus on the classification to provide a conceptual basis for the evaluation of software tools. However, it is a very difficult task to understand what tasks are supported by a software tool in a development lifecycle and to compare it to similar kinds of tools because there are inconsistent definitions for CASE technology. In this section, several classifications of CASE tools are discussed and an activity-based classification is suggested, which provides a much clearer definition of the breadth of CASE tools in the software development phases.

Sommerville addresses the possible classification dimensions such as *Functionality*, *Process Support*, and *Breadth of Support* with which tools can be partitioned into classes for easy assessment and comparison (Ian Sommerville, 1995). Those possible classification dimensions are explained as the following:

Functionality: What functions are supported by tools?

Process Support: What process phases are supported by the CASE technology? Tools can be classified as design tools, programming tools, maintenance tools and so on.

Breadth of Support: What is the extent of process activity supported by the technology? Narrow support means support for very specific tasks in the process such as an entity-relation diagram, compiling a program, etc. Broader support

includes broader support for process phases such as design with the most general support covering all or most phases of the software.

He proposed an activity based classification in Figure 3.7 which shows the process phases supported by a number of different types of CASE tools. While CASE tools for planning and estimating, text editing, document preparation and configuration management may be used throughout the software process, other tools support one or two specific phases in the production lifecycle.

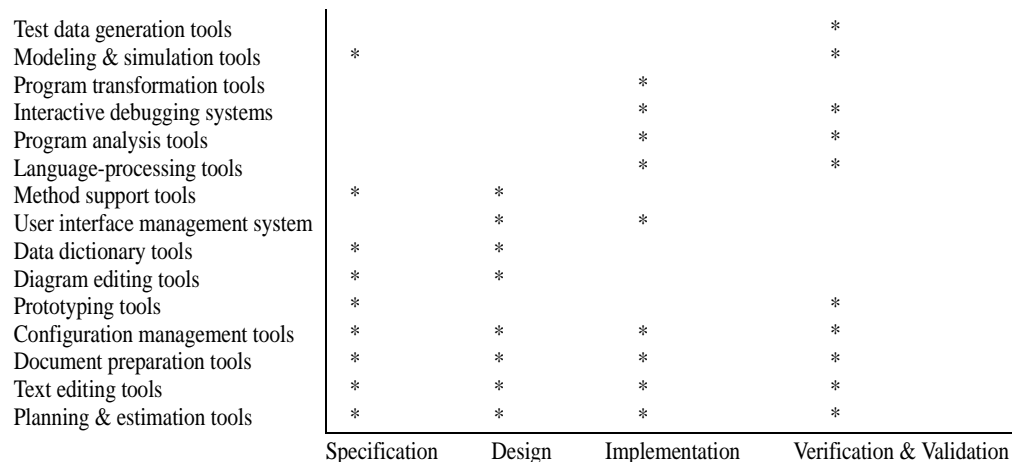


Figure 3.7. Sommerville's Activity-based classification

Forte, G. and McCully, K. (1991) proposed a taxonomy of CASE tools which has two levels . The first level of the taxonomy classifies CASE tools by the domains such as application areas targeted by the tool, tasks supported in the development lifecycle, method and notations, hardware platform, operating systems, databases and transaction processing subsystems supported by tool, etc. The second level specifies attributes for

each domain in two main classes (vertical and horizontal tools) according to development tasks shown in Figure 3.8. While vertical tools support tasks in a specific lifecycle phase or activity, horizontal tools cover tasks throughout the development lifecycle.

Planning	Analysis	Design	Construction	Testing	Maintenance
Project Management					
Process/Quality Assurance					
Workgroup and personal productivity					
CASE environment					
Documentation					

Figure 3.8. Development tasks in Forte and McCully's classification

Pressman suggested a taxonomy of CASE tools which can be categorized by function, by their role as instruments for managers or technical people, by their use in the various steps of the software engineering process, by the environment architecture, and by their origin or cost (Roger S. Pressman, 1992). The categories of the taxonomy are the following:

Business Systems Planning Tools: Tools in this category provide a “Metamodel” from which specific information systems are derived by modeling the strategic information requirements of an organization.

Project Management Tools: This category encompasses CASE tools, Project planning tools, Requirement tracing tools, Metrics tools, Estimation & Scheduling tools, etc.

Support Tools: This category encompasses systems and application tools that complement the software engineering process, Documentation tools, System software tools, Database tools, etc.

Analysis & Design Tools: Tools in this category enable a software engineer to create a model of the system to be built. Those tools are structured analysis/design tools, Prototyping/Simulation tools, Interface Design/Development tools.

Programming Tools: This category encompasses compilers, editor, and debuggers which are available to support most conventional programming languages. In addition, object-oriented programming environments, 4th generation coding tools, application generators, and database query languages also reside within this category.

Integration & Testing Tools: This category includes three sub categories such as static analysis tools, dynamic analysis tools, and test management tools which are most widely used.

Prototyping Tools: Any tool that supports prototyping which is a widely used software engineering paradigm.

Maintenance Tools: The maintenance tools category can be subdivided into the following functions:

Reverse Engineering Tools - takes source code as input and generates graphical structured analysis and design models, where-used lists, and other design information.

Code restructuring and analysis tools - analyze program syntax, generate a control flow graph, and automatically generate a structured program.

Re-engineering tools - used to modify on-line database systems (e.g., IDMS or DB2 files into entity-relationship format)

Framework Tools: Software tools in this category provide database management, configuration management, and CASE tools integration capabilities.

Alfonso Fuggetta (1993) classified CASE products in the product-process technology according to breadth of their support. He partitioned CASE tools into three categories: *Tools*, *Workbenches*, and *Environments*. The first category, *Tools*, support only specific tasks in the software-production process. The classes of the category are shown in Table 3.4.

Class	Subclass
Editing	Graphical editors, Textural editors
Programming	Coding and debugging, Code generators, Code restructurers
Verification & Validation	Static/Dynamic analyzers, Comparators, Symbolic executors, Emulators/Simulators, Correctness proof assistants, Test case generators, Test management tools
Configuration management	Configuration- and version management tools, Configuration builders, Change-control monitors, Librarians
Metrics & measurement	Code analyzers, Execution monitors/timing analyzers
Project management	Cost-estimation tools, Project-planning tools, Conference desk, E-mail, Bulletin boards, Project agenda, Project notebooks
Miscellaneous tools	Hypertext systems, Spreadsheets

Table 3.4. Classes of CASE Tools

The second category, *Workbenches*, support one or a few software process activities by integrating several tools in a single application. This category is partitioned into eight classes of *Workbenches*:

1. Business planning & modeling: This class of *Workbenches* includes products to support the identification and description of a complex business. They are used to build high-level enterprise models to assess the general requirements and information flows, and identify priorities in the development of information systems. (e.g. PC Prism)
2. Analysis & design: Products in this class automate most of the analysis and design methodologies such as structured analysis/design, object-oriented analysis/design, and the Jackson System Development. (e.g. Exelerators, Statemate, StP, etc.)
3. User interface development: This class of *Workbenches* does not help with specific process activities but rather user-interface design and development. (e.g. HP Interface Architect, DEC VUIT)
4. Programming: The *Workbenches* in this class support programming activities including compiling and debugging in a integrated form of a text editor, a compiler/linker, and a debugger. (e.g. CodeCenter)
5. Verification & validation: Products in this *Workbenches* help module and system testing. They integrate tools from both the metrics/measurement class and the verification/validation class. (e.g. Battlemap, ACT, Logiscope, etc.)

6. Maintenance & reverse engineering: Tools used for development are used for maintenance to modify requirement specification, design, and source code. Also, testing procedures are required. Tools for reverse engineering that support the extracting system information and design information out of the existing system are code restructurers, flow charters, and a cross-reference generators. (e.g. Recorder, Rigi, Hindsight, SmartSystems, Ensemble, etc.)
7. Configuration management: The *Workbenches* in this class integrate tools supporting version control, configuration building, and change control. (e.g. HP Apollo, DSEE, SCCS, etc.)
8. Project management: This *Workbenches* includes tools that support specific project-management activities such as planning, task-assignment, and schedule-tracking. (e.g. Coordinator, DEC Plan, Synchronize, etc.)

The last category, *Environments*, supports all or at least a substantial part of the software process with a collection of *Tools* and *Workbenches*. This category is divided into five classes, *Toolkits*, *Language-centered environments*, *Integrated environments*, *Fourth generation environments*, and *Process-centered environments*.

Toolkits are loosely integrated collections of products easily extended by aggregating different tools and workbenches. Toolkits are environments extended from basic sets of operating-system tools. (e.g. Unix Programmer's Work Bench and VMS VAX Set)

Language-centered environments are development environments for specific languages. The main drawback of a language-centered environment is that it may not be feasible to integrate code in different languages. (e.g. Interlisp, Rational, KEE, and Smalltalk)

Integrated environments operate using standard mechanisms with some limitations. All products in an environment share data through the *repository* concept. (e.g. IBM AD/Cycle, DEC Cohesion)

Fourth-generation environments are a subclass of integrated environments and sets of tools and workbenches supporting the development of a specific class of programs such as electronic data processing and business-oriented applications. (e.g. Informix 4GL and Focus)

Process-centered environments are based on formal definitions of software processes and guide development activities by automating process fragments, automatically invoking tools and workbenches, enforcing specific policies, and assisting people in their work. (e.g. East, Enterprise II, Process Wise, etc.)

Sommerville's and Forte & McCully's classifications of CASE tools provide conceptual bases in understanding tool supports of the entire software process. However, they do not take into account mutual dependencies and relationships among CASE tools. Pressman's classification is based on the functionalities supported by CASE tools and does not provide any help in understanding software process supports. This research extends Fuggetta's classification to provide a much clearer conceptual basis in

understanding the breadth of supports which CASE tools offer in a software development process. Figure 3.9 shows the classification of software tools, workbenches, and environments in the current market. It gives a visual coverage of CASE tools in a production process. Unlike Fuggetta's classification, it enumerates general tools and technologies according to the breadth of their support in a lifecycle instead of specific commercial CASE tools.

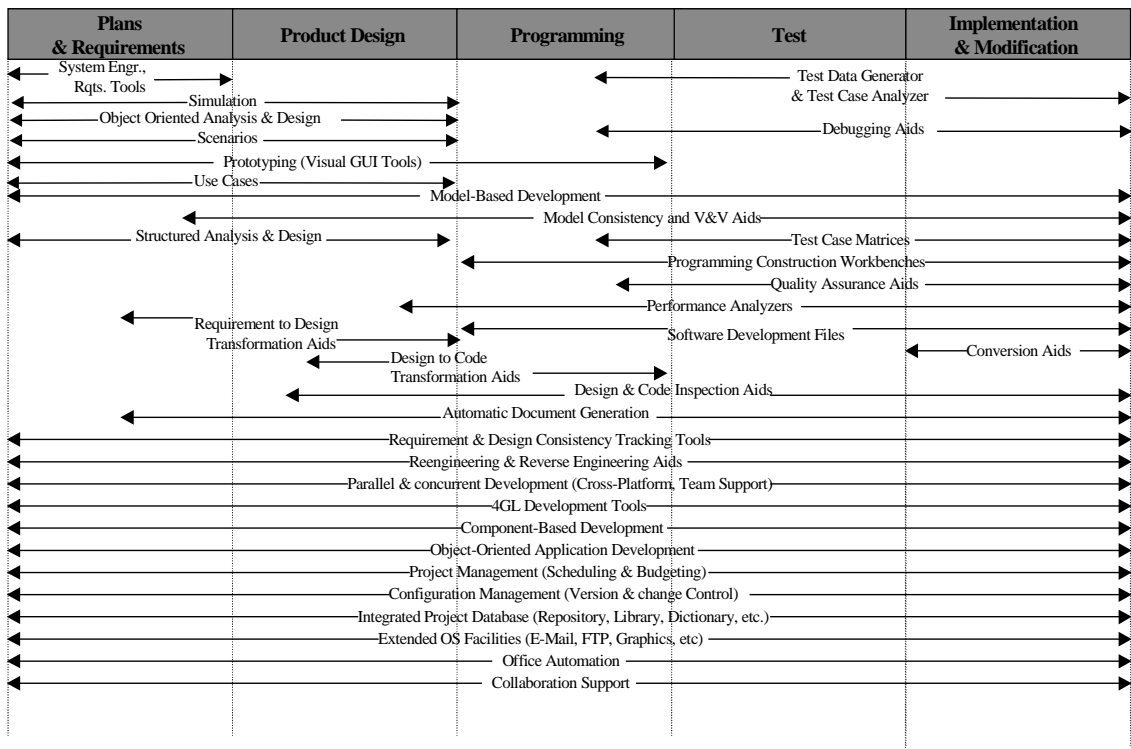


Figure 3.9. Activity Coverage of Software Tools

4.0 Research Approach

This chapter starts with the description of the COCOMO II 7-step modeling methodology which will support the development of the research model that can explain the effect of software tools on the project effort. This chapter suggests three dimensional tool rating scales that reflect the completeness of activity coverage, degree of tool integration, and maturity and user support of tools as an extension of the one dimensional COCOMO II TOOL rating scale. Based on those three tool rating scales, The behavioral analysis results are also described. This chapter also discusses the development of a research model based on the modeling methodology.

4.1 COCOMO II 7-Step Modeling Methodology

This section describes the COCOMO II modeling methodology which will support the process of this research. The 7-step process of the methodology shown in Figure 4.1 has been used to develop COCOMO II and other related models like COQUALMO (COConstructive QUALity MOdel) (Sunita Chulani & Boehm, B. W., 1999). This methodology is adopted to develop the research model that explains the effect of CASE tools on the production process effort.

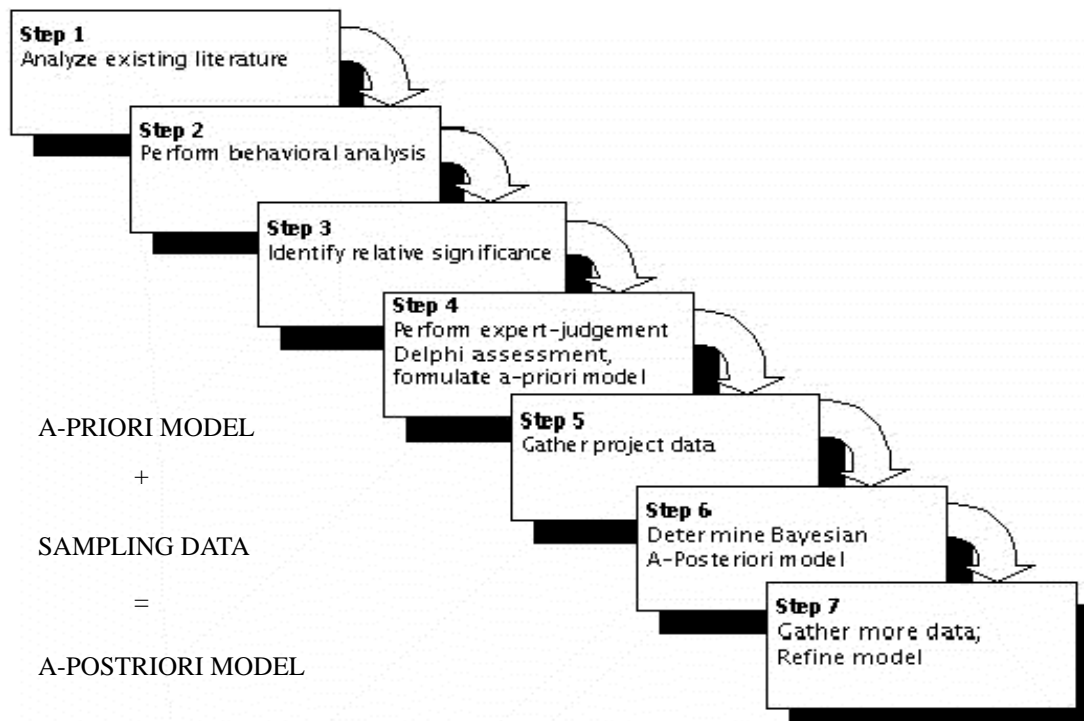


Figure 4.1. The seven-step modeling methodology

Step 1: Analyze literature for factors affecting the quantities to be estimated

The first step in developing a software estimation model is in determining the factors (or predictor variables) that affect the software attribute (or the response variables) being estimated. This was done by reviewing existing literature and analyzing the influence of software tools on development effort. The analysis led to the determination of three relatively orthogonal and potentially significant factors: tool coverage, integration, and maturity/support.

Step 2: Perform behavioral analyses to determine the effect of factor levels on the quantities to be estimated.

This showed the behavioral effects of higher vs. lower levels of each factor on project activity levels.

Step 3: Identify the relative significance of the factors on the quantities to be estimated.

After a thorough study of the behavioral analyses was done, the relative significance and orthogonality of each tool factor was qualitatively confirmed, and rating scales were determined for each factor (Section 4.2). Also, a candidate functional form for a more robust COCOMO II TOOL parameter was determined. This is a weighted average of the three individual ratings:

$$\begin{aligned} \text{TOOL} &= b_1TCOV + b_2TINT + b_3TMAT && \text{(EQ 1)} \\ b_1, b_2, b_3 &\geq 0 \\ b_1 + b_2 + b_3 &= 1 \end{aligned}$$

Step 4: Perform expert-judgement Delphi assessment of quantitative relationships; formulate a-priori version of the model.

Here, cost estimation experts from USC's COCOMO II Affiliates were convened in a two-round Delphi exercise to determine a-priori values of the weight b_1 , b_2 , and b_3 (Section 5.1.1). This model version is based on expert-judgement and is not calibrated against actual project data. But it serves as good starting point as it reflects the knowledge and experience of experts in the field.

Step 5: Gather project data and determine statistical significance of the various parameters.

Fifteen out of 161 projects in the COCOMO II database were found to have valid rating levels for TCOV, TINT, and TMAT. Section 5.1.2 shows the resulting regression analysis.

Step 6: Determine a Bayesian A-Posteriori set of model parameters.

Using the expert-determined drivers and variances as a-priori values, a Bayesian a-posteriori set of model parameters was determined as a weighted average of the a-priori values and the data-determined values and variances, with weights determined by relative variances of the expert and data-based results (Section 5.1.3), including covariance effects.

Step 7: Gather more data to refine model

Continue to gather data, and refine the model to be increasingly data-determined vs. expert-determined.

4.2 Step 1. Tool Rating Scales by Productivity Dimensions

In a software engineering process, the main goal of the use of tools is the improvement of productivity and quality as mentioned earlier. In Chapter 3, what tasks are covered by various tools in the production process was discussed. However, this classification does not explain how much effort can be affected by the use of software tools. Since this research focuses on the impact of software tools on the development effort, this section suggests three different profiles that can evaluate CASE tool by

productivity dimensions such as completeness of tool coverage, degree of tool integration, and tool maturity & user support.

4.2.1 Completeness of Tool Activity Coverage (TCOV)

It is not an easy task to classify CASE tools by their functionalities that support specific tasks in the development process, since a large number of features are offered by current CASE technology (STSC, 1995). However, it is necessary to partition the same kinds of tools by their functionalities to explain differences in the productivity impact. This section provides an extension of the COCOMO TOOL rating scale with currently available CASE tools in the software market.

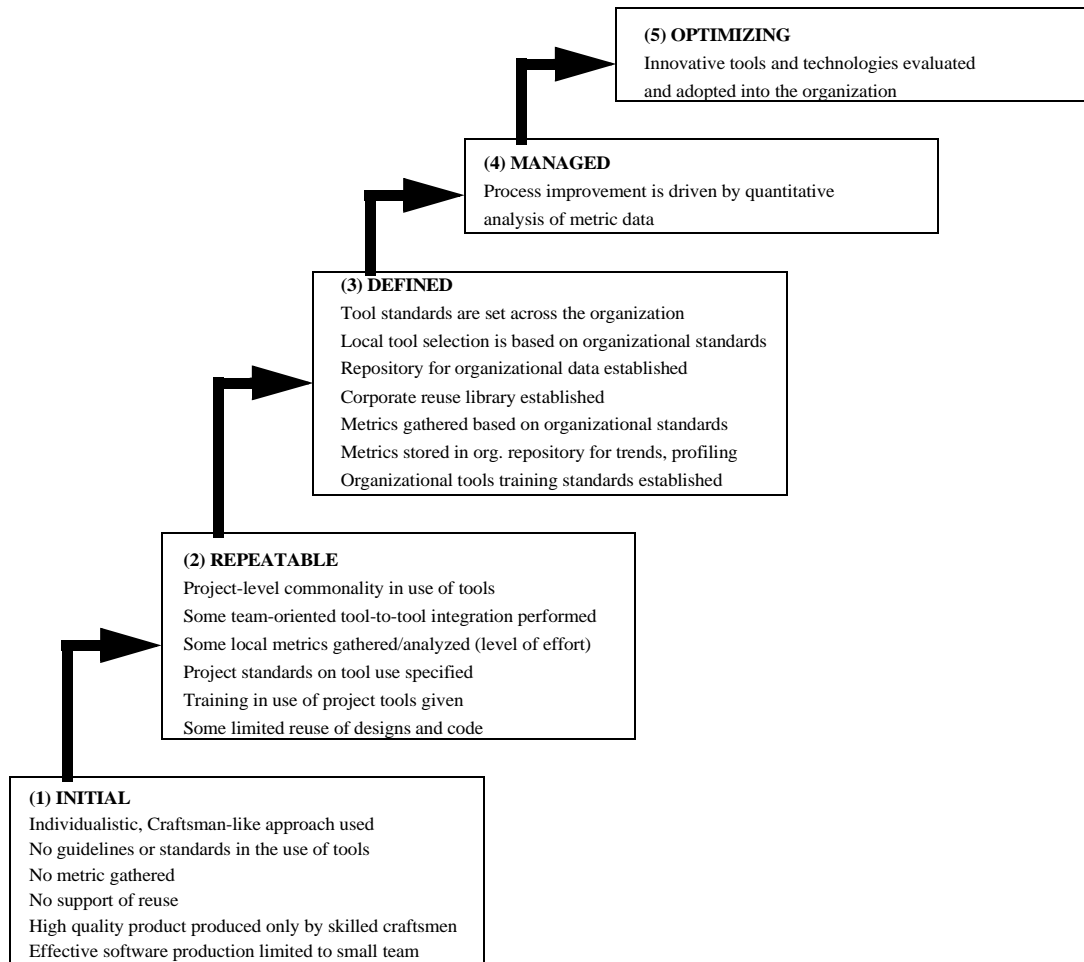


Figure 4.2. Tool-use characteristics at CMM Levels

Alan M. Christie (1993) addressed that there is a clear relationship between tool functionality and the needs of a particular Key Process Area of the Capability Maturity Model that defines five levels of process maturity and provides guidelines to improve the maturity level. Figure 4.2 shows the view of the CASE technology/process relationship. The SEI CMM lists the provision of appropriate resources (including tool support) as a key process area in maturity levels 2 to 5. For instance, if a organization is at level 1, only

limited amount of functionalities may be effectively employed. The higher an organization is in the CMM levels, the much broader set of CASE tools can be applied.

Table 4.1 provides a classification of currently available tool sets based on how completely they support the given activities in the software process and how effectively they support CMM Tool characteristics at each level. Like the COCOMO TOOL rating scale, it also requires a certain amount of judgement to determine an equivalent tool support level.

Rating	TCOV
Very Low	Text-Based Editor Basic 3GL Compiler Basic library Aids Basic Text-based Debugger Basic Linker
Low	Graphical Interactive Editor Simple Design Language Simple Programming Support Library Simple Metrics/Analysis Tool
Nominal	Local Syntax Checking Editor Standard Template Support Document Generator Simple Design Tools Simple Stand-alone Configuration Management Tool Standard Data Transformation Tool Standard Support Metrics Aids with Repository Simple Repository, Basic Test Case Analyzer
High	Local Semantics Checking Editor Automatic Document Generator Requirement Specification Aids and Analyzer Extended Design Tools Automatic Code Generator from Detailed Design Centralized Configuration Management Tool Process Management Aids Partially Associative Repository (Simple Data Model Support) Test Case Analyzer with Spec. Verification Aids Basic Reengineering & Reverse Engineering Tool
Very High	Global Semantics Checking Editor Tailorable Automatic Document Generator Requirement Specification Aids and Analyzer with Tracking Capability Extended Design Tools with Model Verifier Code Generator with Basic Round-Trip Capability Extended Static Analysis Tool Basic Associative, Active Repository (Complex Data Model Support) Heterogeneous N/W Support Distributed Configuration Management Tool Test Case Analyzer with Testing Process Manager, Test Oracle Support Extended Reengineering & Reverse Engineering Tools

Table 4.1. Rating Scale for Completeness of Activity Coverage

Rating	TCOV
Extra High	Groupware systems Distributed Asynchronous Requirement Negotiation and Trade-off tools Code Generator with Extended Round-Trip Capability Extended Associative, Active Repository Spec-based Static and Dynamic Analyzers Pro-active Project decision Assistance

Table 4.1. Rating Scale for Completeness of Activity Coverage (Continued)

4.2.2 Degree of Tool Integration (TINT)

In a software process lifecycle, each individual tool promises higher quality and greater productivity. However, this promise has not been well realized since tool creators can't overcome the difficulties associated with integrating tools in an environment (David Sharon & Rodney Bell, 1995). The main goals of integrating tools are concurrent data sharing and software reuse in a software development lifecycle. If tools are integrated effectively without perturbing other tools in an environment, several benefits such as the reduction of training costs, easier addition of the new system, and software reuse can be obtained (Ian Sommerville, 1995). Thereby, effort and schedule for the software development can be decreased.

Table 4.2 shows another dimension of the CASE tool rating scale to evaluate how well tools are integrated and what kind of mechanisms are provided to exchange information between tools in an integrated environment. This classification is based on Wasserman's five level model for integration of tools in software engineering environments (Wasserman, A. I., 1990). Those five models are:

Platform Integration: Tools run on the same hardware/operating system platform.

Data Integration: Tools operate using the shared data model.

Presentation Integration: Tools offer a common user interface.

Control Integration: Tools may activate and control the operation of other tools.

Process Integration: Tool usage is guided by an explicit process model and associated process engine.

This rating scale classifies CASE tools in the scale, ranging from Very Low to Extra High, according to how well they support the above five integration models and the tool characteristics of the CMM level in Figure 4.2. It provides a set of mechanisms at each level to integrate tools in a development environment. If a different set of mechanisms are used, some amount of judgment is required to fit it to an equivalent level of degree of integration.

Rating	TINT
Very Low	Individual File Formats for Tools (No Conversion Aid), No Activation Control for Other Tools, Different User Interface for each Tool, Fundamental Incompatibilities among Process Assumptions and Object Semantics
Low	Various File Formats for Each Tool (File Conversion Aids), Message Broadcasting to Tools, Some Standardized User Interfaces among Tools, Difficult Incompatibilities among Process Assumptions and Object Semantics
Nominal	Shared-Standard Data Structure, Message Broadcasting through Message Server, Standard User Interface Use among Tools, Reasonably Workable Incompatibilities among Process Assumptions and Object Semantics

Table 4.2. Rating Scale for Degree of Tool Integration

Rating	TINT
High	Shared Repository, Point-to-Point Message Passing, Customizable User Interface Support, Largely Workable Incompatibilities among Process Assumptions and Object Semantics
Very High	Highly Associative Repository, Point-to-Point Message Passing Using reference for Parameters, Some level of Different User Interface, Largely Consistent among Process Assumptions and Object Semantics
Extra High	Distributed-Associative Repository, Extended Point-to-Point Message Passing for Tool Activation, Complete Set of User Interface for Different Level of Users, Fully Consistent among Process Assumptions and Object Semantics (Continued)

Table 4.2. Rating Scale for Degree of Tool Integration

4.2.3 Tool Maturity and User Support (TMAT)

It is very difficult to verify how mature an adopted tool set is for software development. A general way to measure tool maturity is to see how long tools are used in the CASE tool market. As mentioned earlier, the maturity of software tools has a great effect on software quality and productivity. More errors are likely to be introduced during the development with a less mature tool, and consequently more effort is required to correct those errors. Table 4.3 categorizes CASE tools according to the survival length in the software market after they are released.

This rating scale also provides different sets of user support by software vendors. User support is regarded as one of the very important factors to evaluate software tools by Westinghouse's and Ovum's Tool Assessment criteria (Vicky Mosley, 1992; Ovum, 1995). These forms of evaluation criteria evaluates tools quantitatively according to the support by the vendor. Likewise, CASE tools are categorized in a scale, ranging from Very Low to Very High according to their user support.

	Very Low	Low	Nominal	High	Very High	Extra High
Tool Maturity and User Support	Version in pre-release beta-test, Simple documentation and help	Version on market/available less than 6 months, Updated documentation, help available	Version on market/available between 6 months and 1 year, On-line help, tutorial available	Version on market/available between 1 and 2 years, On-line User Support Group	Version on market/available between 2 and 3 years, On-Site Technical User Support Group	Version on market/available more than 3 years

Table 4.3. Rating Scale for Tool Maturity and User Support

4.3 Step 1. Research Model for TOOL Rating Dimensions

Most of software estimation models mentioned earlier are non-linear models and use a multiple regression technique to calibrate the models. This section describes a multiple regression technique and provides a research model based on the multiple regression technique to analyze the TOOL rating dimensions on software effort estimates.

4.3.1 Multiple Regression Analysis

A multiple regression is a methodology that has the form shown in (EQ 12) with a response (estimated) variable and more than one predictor (measured) variables (Weisberg, S., 1985). This methodology is generally used to analyze the linear relationships between the response variable and the multiple predictor variables. The main goal of multiple regression analysis is to understand how the distribution of the response variable y is determined by the possible values of the predictor variables by approximating the true functional relationship with a relatively simple mathematical function. The general form of the multiple regression fitting model is the following.

(EQ 12)

$$y = b_0 + b_1x_1 + b_2x_2 + \dots + b_{k-1}x_{k-1} + b_kx_k + e$$

Where

y: response variable b_0 : intercept
 x_i : i-th predictor variable b_i : coefficient for i-th predictor variable
k: number of the predictor variables e: random error

(EQ 12) is a hyperplane in k-dimensional space for the predictor variable x_i (William W. Hines & Douglas C. Montgomery, 1972). To estimate the unknown regression variables b_i , the method of least squares that minimizes the random error is used. The least square function is as follows:

(EQ 13)

$$L = \sum_{j=1}^n e_j^2 = \sum_{j=1}^n [Y_j - b_0' - b_1(x_{1j} - \bar{x}_1) - \dots - b_k(x_{kj} - \bar{x}_k)]^2$$

Where

n: number of observations e_j : error for j-th observation
 x_{ij} : j-th observation for i-th variable

$$\bar{x}_i = \frac{1}{n} \sum_{j=1}^n x_{ij}$$

$$b_0' = b_0 + b_1\bar{x}_1 + b_2\bar{x}_2 + \dots + b_k\bar{x}_k$$

The number of observations, n, has to be at least k+1 for the regression. By differentiating (EQ 13) and equating to zero for b_0' and b_i yields

(EQ 14)

$$\frac{\partial L}{\partial b_0'} = \sum_{j=1}^n [Y_j - \hat{b}'_0] = 0$$

$$\frac{\partial L}{\partial b_i} = S_{iY} - \hat{b}_1 S_{i1} - \hat{b}_2 S_{i2} - \dots - \hat{b}_k S_{ik} = 0$$

Where

S_{ii} : corrected sum of squares on the i -th predictor variable

$$S_{ii} = \sum_{j=1}^n (x_{ij} - \bar{x}_i)^2$$

S_{rs} : corrected sum of cross products between x_r and x_s

$$S_{rs} = \sum_{j=1}^n (x_{rj} - \bar{x}_r)(x_{sj} - \bar{x}_s)$$

S_{iY} : corrected sum of cross products between x_i and the response variable Y

$$S_{iY} = \sum_{j=1}^n (x_{ij} - \bar{x}_i)(Y_j - \bar{Y})$$

After some algebra, $p = k + 1$ normal equations can be obtained as

(EQ 15)

$$n\hat{b}'_0 = \sum_{j=1}^n Y_j$$

$$\hat{b}_1 S_{i1} + \dots + \hat{b}_k S_{ik} = S_{iY}$$

By solving the equations in (EQ 15), the unknown regression coefficients and intercept can be obtained. These results can be expressed in matrix notations. However, this research does not focus on statistical methods for multiple regression analyses. Detailed information about the multiple regression analysis is available in (William W. Hines & Douglas C. Montgomery, 1972; Weisberg, S., 1985).

As mentioned earlier, most of the software estimation models use non-linear functions to predict effort since linear statistical models are not appropriate to model the production effort with environmental project characteristics. (EQ 16) shows a non-linear, multiplicative model (Bradford Clark, 1997). Since this model can assess the effect of the different input parameters on the final result, this research will use the model to evaluate the impact of tools on software development effort

(EQ 16)

$$\hat{Y} = A \cdot X_1^{B_1} \cdot X_2^{B_2} \cdot \dots \cdot X_k^{B_k}$$

The above non-linear model can be transformed into a linear model by taking the logarithms of both sides. This logarithmic transformation allows the use of regression techniques (W.E. Griffiths, R.C. Hill, & G.G. Judge, 1993). The transformed linear function of (EQ 16) is as follows:

(EQ 17)

$$\log \hat{Y} = B_0 + B_1 \log X_1 + B_2 \log X_2 + \dots + B_k \log X_k$$

In the above equation, B_i describes the percentage change in Y brought about by a percentage change in X_i . Since the complete data for a population is not available in the real world, they can be substituted with the estimates b_i , which can be derived by using the multiple regression approach mentioned above. To perform the regression analysis, the response variable is replaced by the actual observed value Y .

4.3.2 Research Model

COCOMO II has a set of 17 independent variables, called Effort Multipliers, and 5 Scale Factors, mentioned in the previous section. Those variables except TOOL are used as independent variables for the research model. Table 4.4 and Table 4.5 describe the rating scales of the COCOMO II Effort Multipliers. Instead of using the COCOMO II TOOL rating scale, the three extended TOOL rating scales proposed earlier in this paper are used. Each predictor variable has a scale ranging from Very Low to Extra High.

The research model in (EQ 18) is analogous to the COCOMO II Post Architecture Model. However, the TOOL rating value is determined by summarizing the product of each weighting value and rating value as shown in (EQ 19) to reflect the three productivity dimensions of software tools. The main objective of this model is to determine the best-fit of weighting values for the three tool rating scales by combining expert information with sampling information via Bayesian approach. The overall impact of software tools on project effort can be captured by this model.

(EQ 18)

$$Effort = A(Size)^B \left(\prod_{\substack{i=1 \\ i \neq 15}}^{17} EM_i \right) TOOL$$

(EQ 19)

$$TOOL = b_1 \cdot TINT + b_2 \cdot TMAT + b_3 \cdot TCOV$$

$$\sum_{i=1}^3 b_i = 1$$

Sym.	EM	Very Low	Low	Nominal	High	Very High	Extra High
EM ₁	RELY	slight inconvenience	low, easily recoverable losses	Moderate, easily recoverable losses	high financial loss	risk to human life	
EM ₂	DATA		DB bytes/ Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
EM ₃	CPLX	See Table 4.5					
EM ₄	RUSE		none	Across project	Across program	Across product line	Across multiple product lines
EM ₅	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	
EM ₆	TIME			50% use of available execution time	70%	85%	95%
EM ₇	STOR			50% use of available storage	70%	85%	95%
EM ₈	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	
EM ₉	ACAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
EM ₁₀	PCAP	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
EM ₁₁	AEXP	≤ 2 months	6 months	1 year	3 years	6 years	
EM ₁₂	PEXP	≤ 2 months	6 months	1 year	3 years	6 years	
EM ₁₃	LTEX	≤ 2 months	6 months	1 year	3 years	6 years	

Table 4.4. Rating Summary for Effort Multipliers

Sym.	EM	Very Low	Low	Nominal	High	Very High	Extra High
EM ₁₄	PCON	48% / year	24% / year	12% / year	6% /year	3% /year	
EM ₁₅	TOOL	See Table 2.14, 2.15, and 2.16					
EM ₁₆	SITE	International	Multi-city or Multi-company	Same city or metro. area	Same building or complex	Fully collocated	
EM ₁₇	SCED	75% of nominal	85%	100%	130%	160%	

Table 4.4. Rating Summary for Effort Multipliers (Continued)

Complexity is the combined rating of five areas: control operation, computational operations, device-dependent operations, data management operations, and user management operations. The complexity is the subjective average of these five areas.

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Very Low	Straight-line code with a non-nested structured programming operators: DOs, CASEs, IFTHENELSEs. Simple module composition via procedure calls or simple scripts	Evaluation of simple expression: 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., $D=\text{SQRT}(B^{**}2-4.*A*C)$	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.

Table 4.5. Complexity Ratings

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
Nominal	Mostly simple nesting. Some intermodule control. Decision table. 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 structural changes, simple edits. Complex COT-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.
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.

Table 4.5. Complexity Ratings (Continued)

	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations	User Interface Management Operations
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 multi-media virtual reality.

Table 4.5. Complexity Ratings (Continued)

4.4 Hypothesis Testing

In the COCOMO II model, it is assumed that the weighting values for TINT and T_{MAT} are zero. Only the weighting value of TCOV is set to one to derive the TOOL multiplier. The hypothesis for this research is that the benefit of increasing TINT and T_{MAT} also is a reduction in project development effort. Testing for significance of regression in (EQ 20) is accomplished by testing the hypothesis

$$H_0: b_1 = b_2 = b_3 = 0$$

$$H_1: \text{at least one of the above coefficient is not zero}$$

In order to test the above hypotheses, the F-statistic can be used.

$$\left(F_0 = \frac{SS_R/k}{SS_E/(n-k-1)} \right) \sim F_{k, n-k-1}$$

Where

$$SS_{YY} = SS_R + SS_E$$

$$SS_E = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

n: the number of observation

k: the number of predictor variables

If $F_0 > F_{\alpha, k, n-k-1}$, the null hypothesis (H_0) can be rejected. This test can be conveniently done in an analysis of variance (ANOVA) table.

4.5 Step 2 & 3. Behavioral Analysis: Project Activity Differences

This section describes the project activity differences at each level of the three dimensional TOOL rating scales proposed earlier. Table 4.6, Table 4.7, and Table 4.8 indicate the project activity differences resulting from the differing levels of the three productivity dimensions, respectively. In Table 4.6, the larger reduction in effort due to the use of the higher tool set is due partly to the more complete support of the given tasks by causing less errors and more error elimination in the earlier phases. Table 4.7 shows that the larger effort reduction in the higher degree of tool integration is due partly to the more tight integration of software tools that eliminates effort to exchange data, the more automation of tool invocation, the larger levels of user interface support, and so on. Table 4.8 shows the larger effort reduction in the higher mature/support tool level is due to less

error introduction, easier error detection/removal, and higher degree of support for solving problems.

Phase Rating	Plans & Requirements	Product Design	Programming	Test	Implementation & Modification	
Very Low	More effort for simulation, prototyping, requirement tracing, configuration management, document generation, and cooperation/synchronization of team members					
	More requirements errors; harder detection and removal					
	More effort for transforming requirements to design					
	More design errors; harder detection and removal					
	More effort for transforming design to code					
	More code errors; harder detection and removal					
	More effort for generating test cases, capturing project specific metrics, and QA					
					More effort for linking, exercising subsystems, reverse/re-engineering	
	Low	Intermediate level of above effects				
	Nominal	No change				
High	Easier simulation, prototyping, requirement tracing, configuration management, document generation, and cooperation/synchronization of team members					
	Fewer requirements errors; easier detection and removal					
	Less effort for transforming requirements to design					
	Fewer design errors; easier detection and removal					
	Reduced design effort					
	Less effort for transforming design to code					
	Fewer code errors; easier detection and removal					
	Less effort for generating test cases, capturing project specific metrics, and QA					
					Less effort for linking, exercising subsystems, reverse/re-engineering	
	Very High	Higher level of above effects				
Extra High	Highest level of above effects					
	Highly reduced effort for project management					
	Much easier rqt. spec. & update					
	Much easier rqt. spec. & update					
	Highly reduced errors, Much faster detection & removal					
	Much faster communication among team members					

Table 4.6. Project Activity Differences Due to Completeness of Activity Coverage

Phase Rating	Plans & Requirements	Product Design	Programming	Test	Implementation & Modification
Very Low	Weaker navigation through the information maintained by the integrated tools, Weaker control of tool execution, Weaker transition of different user interfaces, More effort for translating incompatible process assumption & object semantics, Weaker data management				
Low	Intermediate level of above efforts				
Nominal	No change				
High	Stronger navigation through the information maintained by the integrated tools, Stronger control of tool execution, Easier transition of different user interfaces, Less effort for translating incompatible process assumption & object semantics, Stronger data management				
Very High	Higher level of above effects				
Extra High	Very strong support for data navigation, data management, user interface transition, tool execution control, translation of incompatible process assumption & object semantics				

Table 4.7. Project Activity Differences Due to Degree of Tool Integration

Phase Rating	Plans & Requirements	Product Design	Programming	Test	Implementation & Modification
Very Low	More errors; harder detection and removal More effort for solving problems due to weak user supports				
Low	Intermediate level of above efforts				
Nominal	No change				
High	Fewer errors; easier detection and removal Less effort for solving problems due to strong user supports				
Very High	Higher level of above effects				
Extra High	Highly reduced errors, Highly reduced effort for detection and removal Highly reduced effort for searching problem solutions				

Table 4.8. Project Activity Differences Due to Tool Maturity and User Support

5.0 Bayesian Approach for Model Calibration

5.1 Bayesian Approach

This chapter describes a bayesian approach that combines two sources of (expert-judgement and data-determined) information to find out the best fit of weighting values for the extended tool rating scales in the research model. It also shows the comparison results of prediction accuracies with the COCOMO II Bayesian effort estimates.

5.1.1 Step 4. Delphi Process Result (A-Priori Model)

To obtain a consensus-based relative weighting value for each of the three TOOL rating scale ratings, two rounds of Delphi analyses were carried out. This Delphi process gives an opportunity to apply the knowledge and experience of experts in the field to the initial model. For this Delphi process, participants were selected from the COCOMO II affiliates such as Commercial, Aerospace, Government, FFRDC, and Consortia organization. The steps taken for the 2-round Delphi process are described below.

First Round:

1. Provide participants with Delphi Questionnaire without initial weighting values
2. Collect responses
3. Ensure validity of responses by correspondence
4. Simple analysis of the responses

Second Round:

1. Provide participants with Delphi Questionnaire with the result of the first round
2. Repeat the above steps (2, 3, and 4)
3. Converge to Final Delphi results

Table 5.1 shows the results of the Delphi process initially carried.

	TCOV	TINT	TMAT
Mean	0.475	0.205	0.32
Variance	0.0025	0.0001	0.0029

Table 5.1. Initial A-Priori Weighting Values

As shown in the above table, the variances of the weighting values are very small. That is, most of the participants agreed to the mean values. Since it causes the problem that all posterior mean values are closer to the prior values when those are combined with sample information. Therefore, another two round of Delphi process was carried out to more strong dispersion of expert inputs on the weighting values. In Table 5.2, the final delphi results are shown and those mean values and variances are used to find out the best-fit posterior weighting values in Bayesian. The participants agreed that the most productivity gains can be obtained via the higher tool ratings in TCOV with 47%. Even though differences in productivity gains in TINT and TMAT are 26% and 27% respectively, the variance of TINT is much smaller than that of TMAT

	TCOV	TINT	TMAT
Mean	0.47	0.26	0.27
Variance	0.025694	0.005485	0.016875

Table 5.2. Final A-Priori Weighting Values

5.1.2 Step 5. Multiple Regression Analysis (Sample)

In order to find out the sampling information on the weighting value of three tool rating scales, 15 projects that have information about TCOV (Completeness of Activity Coverage), TINT (Degree of Tool Integration), and TMAT (Tool Maturity and User Support) were available from the COCOMO 81 database (Boehm, B. W., 1981). Table 5.3 shows the ratings of data from the 15 projects based on the three dimensional TOOL rating scales.

Project No.	TCOV	TINT	TMAT
1	Low	High	High
2	Nominal	Low	Low
3	Nominal	Low	Low
4	Nominal	Low	Very High
5	Nominal	High	Low
6	Nominal	Low	Low
7	Nominal	Low	High
8	Low	High	Very High
9	Nominal	Very Low	Low
10	Low	Low	High
11	Nominal	Low	High

Table 5.3. Three Rating Scale Values from the data of 15 Projects

Project No.	TCOV	TINT	TMAT
12	Nominal	Low	High
13	Nominal	Low	Low
14	Nominal	High	High
15	Very Low	Nominal	Nominal

(Continued)

Table 5.3. Three Rating Scale Values from the data of 15 Projects

The TCOV ratings in the above table are the same as those in the original COCOMO 81 TOOL ratings. Since the data from 15 projects came from 1970's and 1980's, all cost drivers are displaced from COCOMO 81 to COCOMO II. Since the TOOL rating scale is displaced 2 places upward from the original COCOMO 81 to COCOMO II, a transformation table in Table 5.4 is used for the TCOV, TINT, and TMAT ratings.

COCOMO 81 Rating	COCOMO II Multiplier
Very High	1.00
High	1.09
Nominal	1.17
Low	1.29
Very Low	1.45

Table 5.4. TOOL conversion from COCOMO 81 to COCOMO II

In order to determine relative weighting values for TCOV, TINT, and TMAT, the TOOL rating is determined by the regression model shown in (EQ 1).

(EQ 1)

$$TOOL = b_1TCOV + b_2TINT + b_3TMAT$$

Linear regression with the data from the 15 projects by Arc (Dennis Cook & Sanford Weisberg, 1999) gives the following results in Table 5.5.

```

Data set = TOOL_15_data, Name of Fit = L1
Normal Regression
Kernel mean function = Identity
Response      = TOOL
Terms         = (TCOV TINT TMAT)
With no intercept.
Coefficient Estimates
Label      Estimate      Std. Error      t-value
TCOV      0.515982      0.0888635      5.806
TINT      0.282561      0.107657      2.625
TMAT      0.165480      0.111398      1.485

Sigma hat:      0.048214
Number of cases:      15
Degrees of freedom:      12

Summary Analysis of Variance Table
Source      df      SS      MS      F      p-value
Regression  3      20.5056      6.8352      2940.40      0.0000
Residual    12      0.027895      0.00232459

```

Table 5.5. Regression Summary of the Sample

In the above regression result, Estimates are the estimated coefficients for the weighting values of b_1 , b_2 , and b_3 , respectively. The Std. Error is the estimated standard deviation for each coefficient. The t-value is the ratio of residual error and variance for each predictor variable in the regression model and may be interpreted as the signal-to-noise ratio with the corresponding predictor variables. Hence, the higher the t-value, the higher the signal being sent by the predictor variable. In the Summary Analysis of Variance Table, the RSS (Residual Sum of Squares) is 0.027895 with 12 degrees of

freedom. The sum of the estimated coefficients is not 1. Therefore, those values can be normalized. (EQ 2) shows the normalized regression model obtained from 15 project sample.

(EQ 2)

$$TOOL = 0.54 \cdot TINT + 0.29 \cdot TMAT + 0.17 \cdot TCOV$$

5.2 Step 6. Bayesian Analysis (A-Posteriori Model)

Bayesian inference is a statistical method by which similar information is combined to produce a posterior probability distribution of one or more parameters of interest. In Bayesian analysis, probability is defined in terms of a degree of belief and an estimator is chosen in order to minimize expected loss where the expectation is taken with respect to the posterior distribution of unknown parameters (Gerge G. Judge et al., 1985). The procedure that combines A-prior (expert-judged) information with sample information about the parameters of interest to produce a posterior probability distribution is done by using Bayes' theorem as follows:

(EQ 3)

$$g(\theta|y) = \frac{f(y|\theta) \cdot g(\theta)}{f(y)}$$

where

θ : the vector of parameters of interest

y: the vector of sample observations

In the above equation, f denotes a density function for y and g denotes a density function for the unknown parameter vector θ . $f(y|\theta)$ is the joint density function which is algebraically identical to the likelihood function for θ and contains all the sample information about unknown parameter θ . $g(\theta)$ is the prior distribution function for θ

summarizing non-sample information about θ . The posterior distribution function, $g(\theta|y)$, for θ summarizes all information about θ . With respect to θ , $f(y)$ can be regarded as a constant and $f(y|\theta)$ can be written as the likelihood function $l(\theta|y)$, then (EQ 3) can be rewritten as follows:

$$g(\theta|y) \propto l(\theta|y) \cdot g(\theta) \tag{EQ 4}$$

In other words, the above equation can be illustrated as

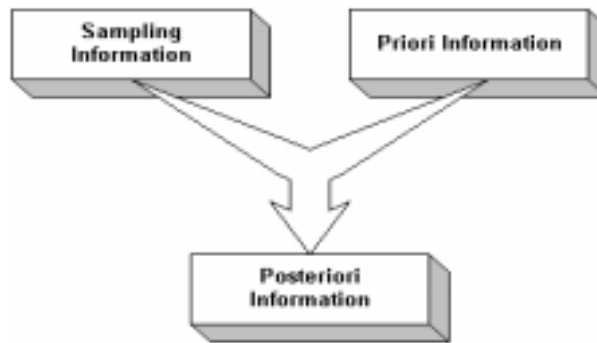


Figure 5.1. Bayesian: Combination of Prior and Sampling Information

The regression model shown in (EQ 1) can be rewritten as

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{e} \tag{EQ 5}$$

where \mathbf{y} is a $(n \times 1)$ vector of sample observation on a response variable, \mathbf{X} is a $(n \times k)$ matrix of observations on k predictor variables, \mathbf{e} is a $(n \times 1)$ disturbance vector with the properties $\mathbf{e} \sim N(\mathbf{0}, \delta^2\mathbf{I})$, and β and δ are unknown parameters in the regression model. thereby, (EQ 4) can be written as follows:

(EQ 6)

$$g(\beta, \delta|y) \propto l(\beta, \delta|y) \cdot g(\beta, \delta)$$

To perform Bayesian analysis on the regression model, A-priori and small sample information is explained in Section 5.1. The posterior mean and variance for the unknown parameters β (or \mathbf{b}^{**}) and δ (or $\text{Var}(\mathbf{b}^{**})$) are defined as follows (Edward E. Leamer, 1978):

(EQ 7)

$$\mathbf{b}^{**} = \left[\frac{1}{s^2} \mathbf{X}'\mathbf{X} + \mathbf{H}^* \right]^{-1} \times \left[\frac{1}{s^2} \mathbf{X}'\mathbf{X}\mathbf{b} + \mathbf{H}^* \mathbf{b}^* \right]$$

$$\text{Var}(\mathbf{b}^{**}) = \left[\frac{1}{s^2} \mathbf{X}'\mathbf{X} + \mathbf{H}^* \right]^{-1}$$

where \mathbf{X} is a matrix of observations on predictor variables, s is the variance of the residual for the sample data, and \mathbf{b}^* and \mathbf{H}^* are the mean of prior information and the inverse of variance matrix, respectively. The posterior mean \mathbf{b}^{**} can be regarded as a matrix average of the prior and sample mean with weight s given by the precisions of all information about the unknown parameters. In other words, if the precision of the prior information is greater than that of the sample information, the posterior values will be closer to the prior values.

The code shown in Table 5.6 was used to obtain the matrices for \mathbf{b}^{**} (denoted as b2) and $\text{Var}(\mathbf{b}^{**})$ (denoted as v2) in Arc.

```

(def x (send L1 :x))
X> (def y (send L1 :y))
Y
> (def xt (transpose x))
XT
> (def xtx (matmult xt x))
XTX
> (def xtxinv (inverse xtx))
XTXINV
> (def xtxinvxt (matmult xtxinv xt))
XTXINVXT
> (def b (matmult xtxinvxt y))
B
> (def bb (bind-rows '(0.47 0.26 0.27)))
BB
> (def v1 (bind-rows '(0.025694 0 0) '(0 0.005485 0) '(0 0 0.016875)))
V1
> (def v1inv (inverse v1))
V1INV
> (def v2inv (+ (matmult (/ 1 (^ (send L1 :sigma-hat) 2)) xtx) v1inv))
V2INV
> (def v2 (inverse v2inv))
V2
> (def xty (matmult xt y))
XTY
> (def ls2xty (matmult (/ 1 (^ (send L1 :sigma-hat) 2)) xty))
1S2XTY
> v1inv
#2A((38.9196 0 0) (0 182.315 0) (0 0 59.2593))
> (def vv1inv (bind-rows '(38.9196 0 0) '(0 182.315 0) '(0 0 59.2593)))
VV1INV
> (def vv1invb1 (matmult vv1inv (bind-columns '(0.47 0.26 0.27))))
VV1INVB1
> (def b2 (matmult (+ ls2xty vv1invb1)))
B2
> b2
(9173.6 9404.04 8798.86)
> (def b2 (matmult v2 (+ ls2xty vv1invb1)))
B2
> b2
(0.495104 0.259691 0.211617)
> v2
#2A((0.00461028 -0.00156482 -0.00307792) (-0.00156482 0.00335019 -
0.00191963) (-0.00307792 -0.00191963 0.00528472))

```

Table 5.6. Bayesian Analysis Procedure in Arc

The derived posterior result for unknown parameters, coefficients and variances from (EQ 7) is summarized with the prior and sample values in Table 5.7. By combining

two sources of information on parameters, the posterior variance gets smaller than those of the prior and sample.

		b₁	b₂	b₃
Prior	Mean	0.47	0.26	0.27
	Variance	0.025694	0.005485	0.016875
Sample	Mean	0.515982	0.282561	0.165480
	Variance	0.0078967	0.011590	0.012409
Posterior	Mean	0.495104	0.259691	0.211617
	Variance	0.00461028	0.00335019	0.00525464

Table 5.7. Point Estimates of Coefficients

As shown in the above table, the sum of posterior means for the weighting values of the extended three TOOL rating scales is not 1. Therefore, TOOL rating value in the research model can be determined by using normalized posterior coefficient means as follows:

(EQ 8)

$$TOOL = 0.51 \cdot TCOV + 0.27 \cdot TINT + 0.22 \cdot TMAT$$

This indicates that differences in tool coverage are the most important determinant of tool productivity gains, with a relative weight of 51%. The next most important is tool integration, with a relative weight of 27%. Tool maturity has the smallest effect but is still significant at a 22% relative weight.

The Research model is calibrated with the data from the 15 projects to determine the best-fit relative weighting values for the three TOOL rating scales. In order to compare the

model with the COCOMO II Bayesian estimation result, an evaluation criterion, the percentage of predictions that fall within X% of the actuals denoted as PRED(X), is used. The models are evaluated at PRED(.10), which is done by counting the number of MRE (Magnitude of Relative Errors) (Conte, S., H. Dunsmore, & V.Shen, 1986) in (EQ 9) less than or equal to 0.10 and dividing by the number of projects. The MRE for each project is defined as the absolute value of the difference between the estimated project effort, $E(Y)$ and actual project effort, Y , relative to the magnitude of the actual effort.

(EQ 9)

$$MRE_i = \left| \frac{E(Y)_i - Y_i}{Y_i} \right|$$

Table 5.8 summarizes the prediction accuracies of COCOMO II, sample without prior information, and posterior with prior information. Even though the prediction accuracies of sample and posterior estimates are the same (87%), the variances of the posterior coefficient estimates are smaller than those of sample regression model as shown in Table 5.7. Both the sample and posterior model give better prediction accuracies than the COCOMO II Bayesian mode does due to 3 dimensional TOOL rating scales.

	COCOMO II Bayesian (1 Dimensional TOOL)	Sample without prior (3 Dimensional TOOL)	Posterior (3 Dimensional TOOL)
PRED (.10)	67%	87%	87%

Table 5.8. Prediction Accuracy Comparison

6.0 Research Model Validation

There are several cross-validation techniques such as *K-fold*, *Leave-one-out*, *Jackknife*, *Delete-d*, and *Bootstrap* for regression problems (Phillip I. Good, 1999). In this chapter, two cross-validation methodologies supported by the program Arc are described. It also show the simulation results with COCOMO II 161 project data used for the calibration of the COCOMO II.2000 (Boehm, B. W. et al., 2000) (with one-dimensional TOOL and with three dimensional TCOV, TINT, and TMAT) to validate the research model.

6.1 Cross-Validation by Data-Splitting

6.1.1 Data Splitting

Cross-Validation by data-splitting is a widely-used model checking method to validate a regression model. This method randomly divides the original dataset into two parts of subsamples - the first part of subsample (*Construction*, *Model-building*, or *Training* subsample) for exploration and model formulation, the second (*Validation* or *Prediction* subsample) for model validation, formal estimation, and testing (John Neter et al., 1996). Although the most ideal validation method is through the collection of new data, it is very hard to get new data in practice. So, data splitting is an attempt to simulate replication of the study.

The validation set is used for the validation of the regression model in the same way as in the newly collected dataset. The regression coefficients are reestimated for the

selected model and then compared for the consistency with the coefficients obtained from the construction subsample. Prediction accuracy can be determined from the data in the validation subsample from the regression model built from the construction subsample. A possible drawback of the cross-validation by splitting the original dataset is that the variance of the estimated regression model from the construction dataset usually gets larger than the variance obtained from the regression model with the entire dataset.

In order to diagnose the cross-validation method, the predicted residual sum of squares (or PRESS) shown in (EQ 1) is used as a criterion function.

$$PRESS = \sum \hat{e}_{(i)}^2 \tag{EQ 1}$$

where

$$\hat{e}_{(i)}^2 = y_i - X_i^T \hat{\beta}_{(i)} = \frac{\hat{e}_i}{1 - h_{ii}}$$

If PRESS is small, it indicates that the estimated regression model is a good model. Another criterion function is R^2 as shown in (EQ 2). If R^2 is 1, all data in the construction set must fall exactly on a straight line with no variance.

$$R^2 = 1 - \frac{nPRESS}{SST} \tag{EQ 2}$$

6.1.2 Cross-Validation in Arc

The Program Arc supports a very simple cross-validation by splitting the original dataset. The basic outline used in Arc is (Sanford Weisberg, 1999):

1. Divide the data into two parts, a “construction” set and a “validation” set.

2. Fit a model of the interest to the construction set.
3. Compute a summary statistic, usually a function of the deviance and possibly the number of parameters, based on the validation set only.
4. Select models that make the statistic chosen small (usually) when applied to the validation data.

In Arc's cross-validation, the fitted model excludes the validation from the fitting. The output summary displays the weight sum of squared deviations and the number of completed observations in the validation set.

6.1.3 Cross-Validation Results Using Arc

As mentioned earlier, the sample distribution of the COCOMO II.2000 parameters was calculated with 161 project data under the log-transformed linear regression model as the following.

(EQ 3)

$$\log PM = \beta_0 \log A + \beta_1 B \log Size + \beta_2 SF_1 \log Size + \dots + \beta_6 SF_5 \log Size + \beta_7 \log Size EM_1 + \beta_8 \log Size EM_2 + \dots + \beta_{23} \log Size EM_{17}$$

In order to validate the model, two dataset of the same size (161 project data) were used. For the TOOL rating value of the first dataset, a one dimensional TOOL rating scale is used. The TOOL rating value of the 15 project data used for Bayesian in the second dataset is determined by a weighted sum of the products of the weighting value obtained by Bayesian analysis and the extended TOOL ratings (TCOV, TIN, and TMAT). The

TOOL ratings of the rest of the project data in the second dataset have the same values in the first dataset.

For the first experiment, 15 project data that has information about the three TOOL ratings are intentionally assigned to the validation dataset. For the exclusion of those data from the construction set, they are moved to the top of the dataset and then removed by using the following command:

```
>(send COCOMOII_1_TOOL :cross-validation :new t :validation-set (iseq 15))
```

In Arc, the above command sets a cross-validation by splitting the original dataset (COCOMOII_1_TOOL) into two subsets in which the validation set is composed of the first 15 project data. The arguments in the above command are explained as follows:

:new - New must be set to t to start cross-validation. If :new is not set, then the case numbers in the validation set are displayed.

:validation-set - An optional list of case numbers to be put in the validation set

iseq 15 - select cases (case numbers 0 to 14) from the dataset

Since the observations in the two construction sets are the same, the regression coefficients obtained from the construction sets by OLS (Ordinary Linear Square) in Arc are the same as shown in Table 6.1. However, the TOOL ratings in the two validation sets are different as mentioned earlier. The TOOL ratings for the observations in the first validation set are the ratings determined by the one-dimensional TOOL, which is the same as in COCOMO II.2000. The TOOL ratings for the observations in the second validation set are determined by the extended three-dimensional TCOV, TINT, and TMAT.

```

Data set = COCOMOII_TOOL, Name of Fit = L1
Deleted cases are
(0 1 2 3 4 5 6 7 8 9 10 11 12 13 14)
Normal Regression
Kernel mean function = Identity
Response      = log[PM]
Terms         = (LOG[SIZE] PMATXLOG[SIZE] PRECXLOG[SIZE] RESLXLOG[SIZE] FLEXX-
LOG[SIZE] TEAMXLOG[SIZE] log[ACAP] log[AEXP] log[CPLX] log[DATA] log[DOCU]
log[LTEX] log[PCAP] log[PCON] log[PEXP] log[PVOL] log[RELY] log[RUSE]
log[SCED] log[SITE] log[STOR] log[TIME] log[TOOL])
Coefficient Estimates
Label          Estimate      Std. Error    t-value
Constant       0.935116      0.112094     8.342
LOG[SIZE]      2.15392      0.117704    18.299
PMATXLOG[SIZE] 0.0168816    0.0137407    1.229
PRECXLOG[SIZE] 0.0317152    0.0118288    2.681
RESLXLOG[SIZE] 0.0310717    0.0154430    2.012
FLEXXLOG[SIZE] -0.00147026   0.0178473   -0.082
TEAMXLOG[SIZE] 0.0107892    0.0194624    0.554
log[ACAP]      0.940347     0.333022     2.824
log[AEXP]      0.230921     0.383999     0.601
log[CPLX]      1.02384     0.247269     4.141
log[DATA]      0.751994     0.246248     3.054
log[DOCU]      1.88080     0.571091     3.293
log[LTEX]      0.0664662    0.474326     0.140
log[PCAP]      1.37783     0.364579     3.779
log[PCON]      0.657545     0.402296     1.634
log[PEXP]      0.809292     0.500946     1.616
log[PVOL]      1.09463     0.309514     3.537
log[RELY]      0.867786     0.383227     2.264
log[RUSE]      -1.16099     0.686944    -1.690
log[SCED]      1.26662     0.331924     3.816
log[SITE]      0.702753     0.568178     1.237
log[STOR]      0.540300     0.438909     1.231
log[TIME]      1.94306     0.451107     4.307
log[TOOL]      0.940123     0.386596     2.432

R Squared:          0.964438
Sigma hat:          0.326693
Number of cases:    161
Number of cases used: 146
Degrees of freedom: 122

Summary Analysis of Variance Table
Source      df      SS      MS      F      p-value
Regression  23     353.124  15.3532  143.85  0.0000
Residual    122    13.0209  0.106728

```

Table 6.1. Fit for the construction of 146 project data

In Table 6.2, the cross-validation summaries and PRESS for the two simulations are displayed. The output for the cross-validation in Arc includes a few summary statistics such as the weighted sum of squared deviations, mean of squared deviations, and a number of observations in the validation set. The sum of squared deviations, 1.14945, and PRESS (0.187743) in the cross-validation using the extended three dimensional TOOL rating scales are smaller than the sum of squared deviations (1.29381) and PRESS (0.189826) in the cross-validation using the one dimensional TOOL rating scale. That is, the observations in the second validation set is a better fit to the regression model obtained from the same construction set.

<p>1st validation set (TOOL)</p>	<p>Cross validation summary of cases not used to get estimates: Sum of squared deviations: 1.29381 Mean squared deviation: 0.0862541 Sqrt(mean squared deviation): 0.29369 Number of observations: 15</p> <p>>(/ (sum (^ (/ (send L1 :residuals) (- 1 (send L1 :leverages)))) 2)) (send L1 :num-included)) 0.189826</p>
<p>2nd validation set (TCOV, TINT, and TMAT)</p>	<p>Cross validation summary of cases not used to get estimates: Sum of squared deviations: 1.14945 Mean squared deviation: 0.0766299 Sqrt(mean squared deviation): 0.276821 Number of observations: 15</p> <p>>(/ (sum (^ (/ (send L1 :residuals) (- 1 (send L1 :leverages)))) 2)) (send L1 :num-included)) 0.187743</p>

Table 6.2. Cross Validation Summary and PRESS

In the second experiment for the cross validation, the same datasets in the first experiment are used except that the first validation set are selected at random with probability fraction set by the following command.

```
>(send COCOMOII_TOOL :cross-validation :new t :fraction .3)
```

:fraction - The approximated fraction of cases to be put into the validation set. For each case, a Bernoulli random deviate with probability :fraction of success is generated. All cases with the generated value 1 are put in the validation set.

Table 6.3 shows the regression fit with the dataset in which TOOL rating are determined by the one dimensional rating scale. For the regression fit, 115 randomly chosen project data is used. 46 project data is assigned into the validation set. In Table 6.4, Cross-validation summary and PRESS are displayed.

```

Data set = COCOMOII_1_TOOL, Name of Fit = L1
46 cases have been deleted.
Normal Regression
Kernel mean function = Identity
Response      = log[PM]
Terms         = (LOG[SIZE] PMATXLOG[SIZE] PRECXLOG[SIZE] RESLXLOG[SIZE]
FLEXXLOG[SIZE] TEAMXLOG[SIZE] log[ACAP] log[AEXP] log[CPLX] log[DATA]
log[DOCU] log[LTEX] log[PCAP] log[PCON] log[PEXP] log[PVOL] log[RELY]
log[RUSE] log[SCED] log[SITE] log[STOR] log[TIME] log[TOOL])
Coefficient Estimates
Label          Estimate      Std. Error   t-value
Constant      0.857377      0.126932     6.755
LOG[SIZE]     2.20778      0.125668    17.568
PMATXLOG[SIZE] 0.0136270    0.0147642    0.923
PRECXLOG[SIZE] 0.0458109    0.0123268    3.716
RESLXLOG[SIZE] 0.0328329    0.0170526    1.925
FLEXXLOG[SIZE] -0.0178236   0.0180021   -0.990
TEAMXLOG[SIZE] 0.00340669   0.0209328    0.163
log[ACAP]     0.827577     0.350431     2.362
log[AEXP]     0.289507     0.393476     0.736
log[CPLX]     0.903131     0.260106     3.472
log[DATA]     0.725348     0.274620     2.641
log[DOCU]     2.29413      0.591367     3.879
log[LTEX]     -0.304636    0.528984    -0.576
log[PCAP]     1.37407      0.390746     3.517
log[PCON]     0.700669     0.398075     1.760
log[PEXP]     0.845777     0.503937     1.678
log[PVOL]     0.785673     0.346415     2.268
log[RELY]     1.37687      0.375584     3.666
log[RUSE]     -1.38177     0.723777    -1.909
log[SCED]     1.10106      0.302119     3.644
log[SITE]     0.907612     0.608789     1.491
log[STOR]     2.29151      0.600632     3.815
log[TIME]     1.44740      0.486658     2.974
log[TOOL]     0.607863     0.391255     1.554

R Squared:          0.967696
Sigma hat:          0.296881
Number of cases:    161
Number of cases used: 115
Degrees of freedom: 91

Summary Analysis of Variance Table
Source      df      SS      MS      F      p-value
Regression  23    240.267  10.4464  118.52  0.0000
Residual    91     8.02061  0.0881386

```

Table 6.3. Fit for the construction set using one dimensional TOOL

```

Cross validation summary of cases not used to get estimates:
Sum of squared deviations:      9.76793
Mean squared deviation:        0.212346
Sqrt(mean squared deviation):  0.460811
Number of observations:        46

> (/ (sum (^ (/ (send L1 :residuals) (- 1 (send L1 :leverages))) 2))
(send L1 :num-included))
0.262203

```

Table 6.4. Cross Validation Summary and PRESS for one dimensional TOOL validation set

In the second dataset for the cross validation, the TOOL ratings of data from 15 out of 161 projects were determined as:

$$\text{TOOL} = 0.51 * \text{TCOV} + 0.27 * \text{TINT} + 0.22 * \text{TMAT}$$

In order to compare with the regression fit with the first dataset, the same observations are assigned into the construction set and the validation set except that the TOOL ratings of the 15 project data are different from those in the first dataset. Table 6.5 shows the regression fit obtained from 115 observations in the construction set. Of those 15 projects, 11 were in the construction set and 4 were in the validation set.

```

Data set = COCOMOII_3_TOOL, Name of Fit = L1
46 cases have been deleted.
Normal Regression
Kernel mean function = Identity
Response      = log[PM]
Terms         = (LOG[SIZE] PMATXLOG[SIZE] PRECXLOG[SIZE] RESLXLOG[SIZE]
FLEXXLOG[SIZE] TEAMXLOG[SIZE] log[ACAP] log[AEXP] log[CPLX] log[DATA]
log[DOCU] log[LTEX] log[PCAP] log[PCON] log[PEXP] log[PVOL] log[RELY]
log[RUSE] log[SCED] log[SITE] log[STOR] log[TIME] log[TOOL])
Coefficient Estimates
Label          Estimate      Std. Error   t-value
Constant      0.848170      0.126720     6.693
LOG[SIZE]     2.21762      0.125242    17.707
PMATXLOG[SIZE] 0.0122937    0.0147512    0.833
PRECXLOG[SIZE] 0.0448272    0.0123358    3.634
RESLXLOG[SIZE] 0.0337900    0.0170215    1.985
FLEXXLOG[SIZE] -0.0185370   0.0179524   -1.033
TEAMXLOG[SIZE] 0.00431776   0.0208810    0.207
log[ACAP]     0.817284     0.347923     2.349
log[AEXP]     0.322897     0.393903     0.820
log[CPLX]     0.911352     0.258616     3.524
log[DATA]     0.714569     0.273790     2.610
log[DOCU]     2.29158      0.589112     3.890
log[LTEX]     -0.305819    0.526855    -0.580
log[PCAP]     1.38206      0.388702     3.556
log[PCON]     0.686581     0.396206     1.733
log[PEXP]     0.776759     0.508492     1.528
log[PVOL]     0.800612     0.344637     2.323
log[RELY]     1.37111      0.374211     3.664
log[RUSE]     -1.35344     0.721037    -1.877
log[SCED]     1.08515      0.301545     3.599
log[SITE]     0.953494     0.608734     1.566
log[STOR]     2.27612      0.598619     3.802
log[TIME]     1.43499      0.485027     2.959
log[TOOL]     0.672681     0.380138     1.770

R Squared:          0.967943
Sigma hat:          0.295748
Number of cases:    161
Number of cases used: 115
Degrees of freedom: 91

Summary Analysis of Variance Table
Source      df      SS      MS      F      p-value
Regression  23    240.328  10.449  119.46  0.0000
Residual    91     7.95946  0.0874666

```

Table 6.5. Fit for the construction set using three dimensional TOOL

Table 6.6 shows the cross validation summary and PRESS from 46 validation observations which are the same as the case in Table 6.4. However, some of the observations in the second validation set have the TOOL ratings by using the extended tool rating scales. The sum of squared deviations (9.7475) with three dimensional TOOL validation set is a little bit smaller than the sum of squared deviations (9.76793) with the one dimensional TOOL validation set. The PRESS (0.260493) in Table 6.6 is also smaller than the PRESS (0.262203) in Table 6.4. Therefore, a few better regression models can be found by using the extended three dimensional TOOL rating scales. The differences are not great, as only 9% (15 of 161) project data points have different values. But, they indicate an improvement, and they indicate stability with respect to the other variables.

```

Cross validation summary of cases not used to get estimates:
Sum of squared deviations:      9.7475
Mean squared deviation:        0.211902
Sqrt(mean squared deviation):  0.460328
Number of observations:        46

>(/ (sum (^ (/ (send L1 :residuals) (- 1 (send L1 :leverages))) 2))
(send L1 :num-included))
0.260493

```

Table 6.6. Cross Validation Summary and PRESS for three dimensional TOOL validation set

6.2 Cross-Validation by Bootstrap

6.2.1 Bootstrap

The bootstrap is a statistical simulation methodology that resamples from the original data set (Michael R. Chernick, 1999). Initiatives of this methodology were done to solve two of the most important problems (the determination of an estimator for a

particular parameter of interest and the evaluation of that estimator through estimates of the standard error of estimator and the determination of confidence intervals) in applied statistics. Because of its generality, it has been used to wider application areas such as nonlinear regression, logistics regression, spatial modeling, and so on.

Here is a formal definition of the bootstrap by Bradley Efron & Robert J. Tibshirani, Efron (1993): Given a sample that has n independent identically distributed random vectors $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \dots, \mathbf{X}_n$ and a real-valued estimator $\theta(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \dots, \mathbf{X}_n)$ (denoted by $\hat{\theta}$), the procedure for the bootstrap to assess the accuracy of $\hat{\theta}$ is defined in terms of empirical distribution function F_n that assigns the same probability $1/n$ to each observed value of the random vectors \mathbf{X}_i ($i = 1, 2, \dots, n$) and is the likelihood estimator of the distribution for the observations when no parametric assumption are made. The bootstrap estimates is denoted θ^* that contains information to be used in making inference from data. The bootstrap distribution for $\hat{\theta} - \theta$ can be obtained by generating $\hat{\theta}$ values by sampling independently with replacement from empirical distribution F_n . The bootstrap estimation of standard error ($\theta^* - \hat{\theta}$) is the standard deviation of the bootstrap distribution for $\hat{\theta} - \theta$.

From the bootstrap sampling, a Monte Carlo approximation of the bootstrap estimates is obtained. The procedure is described below:

1. Generate a bootstrap sample of size n (where n is the original sample size) with replacement from the original distribution.

2. Compute θ^* , the value of $\hat{\theta}$ obtained by using the bootstrap sample in place of the original sample.
3. Repeat steps 1 and 2, k times.

This provides a Monte Carlo approximation to the distribution of θ^* . The standard deviation of the Monte Carlo distribution of θ^* is the Monte Carlo approximation of the bootstrap estimate of the standard error for $\hat{\theta}$. As the number of times of repeating steps 1 and 2 gets larger, there is very little difference between the bootstrap estimator and the Monte Carlo approximation. The main idea of the boot strap is that two distributions are expected to be nearly the same. That means the distribution of $\hat{\theta} - \theta$ behaves like the distribution of $\theta^* - \hat{\theta}$.

In a few cases, the bootstrap estimator can be directly computed without Monte Carlo approximation. (EQ 4) shows how to compute the bootstrap estimate of the standard error of θ .

(EQ 4)

$$\hat{\delta}_{boot} = [(n-1)/n]^{1/2} \hat{\delta}$$

Where

$$\hat{\delta} = \left[\frac{1}{n(n-1)} \sum_{i=1}^n (x_i - \bar{x})^2 \right]^{1/2}$$

x_i : the value of i^{th} observation

\bar{x} : the mean of the sample

6.2.2 Bootstrap in Arc

For bootstrap sampling, the program Arc provides two paradigms to resample with replacement from the original sample (Iain Pardoe, 2000). One way is the “resampling residuals” paradigm that takes the point of view that responses, $y|\mathbf{x}$, are sampled from a univariate distribution, $F(y|\mathbf{x})$. The mean and variance of $F(y|\mathbf{x})$ are given by the following mean function and variance function:

(EQ 5)

$$E(y|\mathcal{X}) = E(y|\mathcal{U}) = \theta_0 + \theta_1 u_1 + \dots + \theta_{k-1} u_{k-1} = \theta^T \mathcal{U}$$
$$Var(y|\mathcal{X}) = \delta^2/w$$

Where

\mathcal{U} : $k \times 1$ matrix driven from \mathbf{x} , all elements are constant 1

θ^T : $k \times 1$ vector of mean function coefficients

w : weight > 0

The above mean and variance function assume that $E(e) = 0$, $Var(e) = \delta^2$ and the distribution errors are independent of \mathbf{x} . Estimation of $F(y|\mathbf{x})$ does not work directly because of conditioning on \mathbf{x} . Since \mathbf{x} is assumed to be fixed under this paradigm, Arc estimate the distribution of e by resampling residuals, defined to be weighted differences between the observed values of the response and the fitted values under the linear regression model. The weighted differences can be written as:

(EQ 6)

$$\hat{e} = \sqrt{w}(y - \hat{E}(y|\mathcal{X})) = \sqrt{w}(y - \hat{y})$$

The bootstrap method by resampling residuals assumes that the linear regression holds.

The other way is the “resampling cases” paradigm that takes the point of view cases, (y, \mathbf{x}) , are sampled from a multivariate distribution $F(y, \mathbf{x})$. This bootstrap method in Arc is very straightforward and estimate the distribution of (y, \mathbf{x}) by resampling cases, defined to be the multivariate vectors (y, \mathbf{x}) . It does not make any assumption about whether or not the linear regression holds. The detailed information about the bootstrap methods supported by Arc is available in (Iain Pardoe, 2000).

6.2.3 Bootstrap Results Using Arc

As shown in (EQ 3), COCOMO II uses a log-transformed regression model sampled from a multivariate distribution, $F(y, \mathbf{x})$. In order to estimate the distribution of (y, \mathbf{x}) by using the resampling cases paradigm in Arc, the same datasets used in cross-validation by data splitting are used. As mentioned earlier, the TOOL ratings of all observations in the first dataset are determined by the one-dimensional TOOL ratings and the TOOL ratings for only data from 15 out of 161 project data points in the second dataset are the sum of the products of the weighting values and the extended TOOL ratings.

```

Data set = COCOMOII_1_TOOL, Name of Fit = L1
Normal Regression
Kernel mean function = Identity
Response      = log[PM]
Terms        = (LOG[SIZE] PMATXLOG[SIZE] PRECXLOG[SIZE] RESLX-
LOG[SIZE] FLEXXLOG[SIZE] TEAMXLOG[SIZE] log[ACAP] log[AEXP] log[CPLX]
log[DATA] log[DOCU] log[LTEX] log[PCAP] log[PCON] log[PEXP] log[PVOL]
log[RELY] log[RUSE] log[SCED] log[SITE] log[STOR] log[TIME]
log[TOOL])
Coefficient Estimates

```

Label	Estimate	Std. Error	t-value
Constant	0.952642	0.106085	8.980
LOG[SIZE]	2.13567	0.110230	19.375
PMATXLOG[SIZE]	0.0175983	0.0130047	1.353
PRECXLOG[SIZE]	0.0288050	0.0109798	2.623
RESLXLOG[SIZE]	0.0312713	0.0146951	2.128
FLEXXLOG[SIZE]	0.00853516	0.0166122	0.514
TEAMXLOG[SIZE]	0.00896105	0.0185281	0.484
log[ACAP]	1.05512	0.310897	3.394
log[AEXP]	0.120875	0.356859	0.339
log[CPLX]	0.954477	0.222730	4.285
log[DATA]	0.800343	0.232736	3.439
log[DOCU]	1.84820	0.552754	3.344
log[LTEX]	0.137340	0.442661	0.310
log[PCAP]	1.30406	0.334579	3.898
log[PCON]	0.645009	0.373569	1.727
log[PEXP]	0.966785	0.440865	2.193
log[PVOL]	1.23328	0.289703	4.257
log[RELY]	0.931019	0.347286	2.681
log[RUSE]	-0.722342	0.609932	-1.184
log[SCED]	1.22439	0.281818	4.345
log[SITE]	0.670258	0.531335	1.261
log[STOR]	0.935885	0.374885	2.496
log[TIME]	1.51208	0.368245	4.106
log[TOOL]	0.929203	0.367768	2.527

```

R Squared:          0.963663
Sigma hat:          0.318667
Number of cases:    161
Degrees of freedom: 137

Summary Analysis of Variance Table

```

Source	df	SS	MS	F	p-value
Regression	23	368.948	16.0412	157.97	0.0000
Residual	137	13.9122	0.101549		

Table 6.7. Fit of 161 project data (One-dimensional TOOL)

Table 6.7 shows the regression fit by using the first dataset with the one dimension TOOL rating scale. To begin a bootstrap analysis by using resampling cases from the above fit, the following command can be used.

```
> (def c_1 (send L1 :bootstrap :nboots 1000 :method :c-boot))
```

:nboots - number of simulations

:method - bootstrap method

c-boot: resampling cases

r-boot: resampling residuals

This command creates a bootstrap object *c* to be used for the “resampling cases” paradigm. It makes 1000 boots that have the same size as that of the original dataset (161 project data). In order to get information about the bootstrap object, *c*, typed commands can be used. For example, the following typed command creates the histograms for the coefficient estimates.

```
> (send c_1 :histograms)
```

Figure 6.1 shows the histograms for Intercept and log[TOOL]. All histograms for the coefficient estimates are available in Appendix B. The two histograms with superimposed Gaussian kernel density smooths indicates that there is a near-normality

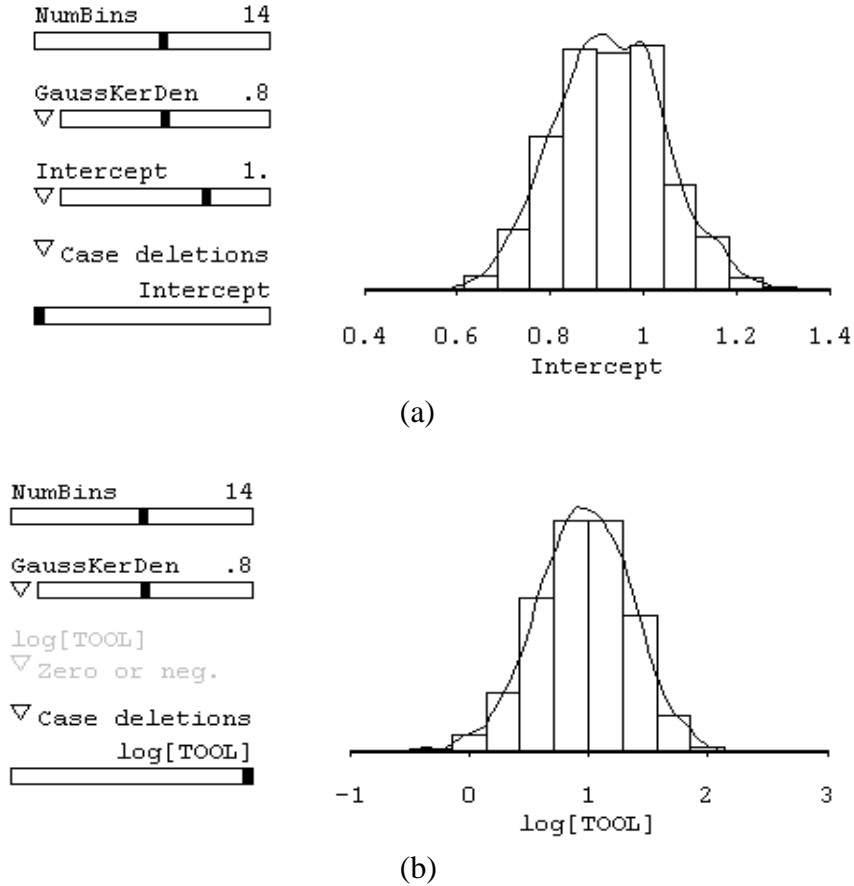


Figure 6.1. Histograms for Intercept and log[TOOL]

Like the above command to get a histograms, the following command produces a probability plot of the *studentized* bootstrap estimates with a t distribution on $n - p$ degree of freedom.

```
> (send c1 :probability-plots)
```

Here is the definition of the *studentized* bootstrap slope estimates that comes from

the “ $\hat{\theta}^*$ is to $\hat{\theta}$ as $\hat{\theta}$ is to θ ” idea.

$$(\hat{\theta}_0^* - \hat{\theta}_0) / \text{se}(\hat{\theta}_0^*)$$

where

$\hat{\theta}_0^*$: the bootstrap estimate of θ_0 and $\hat{\theta}_0$: the usual estimate of θ_0

$se(\hat{\theta}_0^*)$: the bootstrap estimate of the standard error of $\hat{\theta}_0$

Two probability plots for Intercept and log[TOOL] shown in Figure 6.2 with superimposed lines indicate that there is near-normality. The points in the two plots are close to a straight line. All probability plots are also available in Appendix B.

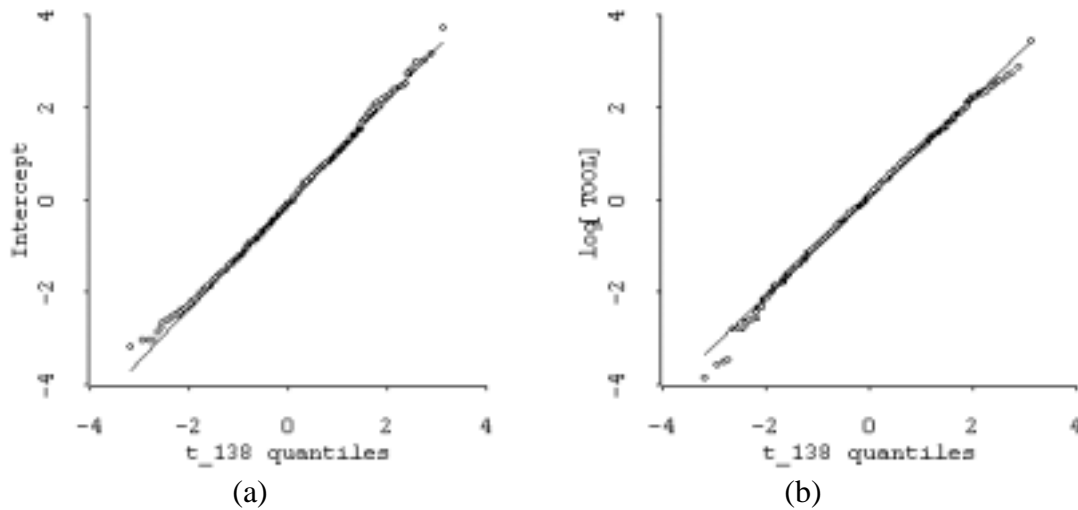


Figure 6.2. Probability Plots for Intercept and log[TOOL]

Another bootstrap object is created by using the resampling cases method in Arc with the regression fit shown in Table 6.8 obtained from the second dataset in which the TOOL ratings of data from 15 out of 161 projects are determined by using the extended three TOOL rating scales. The histograms and the probability plots obtained from the bootstrap object are also displayed in Appendix B indicating that there is also near-normality.

```

Data set = COCOMOII_3_TOOL, Name of Fit = L1
Normal Regression
Kernel mean function = Identity
Response      = log[PM]
Terms         = (LOG[SIZE] PMATXLOG[SIZE] PRECXLOG[SIZE] RESLXLOG[SIZE]
FLEXXLOG[SIZE] TEAMXLOG[SIZE] log[ACAP] log[AEXP] log[CPLX] log[DATA]
log[DOCU] log[LTEX] log[PCAP] log[PCON] log[PEXP] log[PVOL] log[RELY]
log[RUSE] log[SCED] log[SITE] log[STOR] log[TIME] log[TOOL])
Coefficient Estimates
Label          Estimate      Std. Error   t-value
Constant      0.944937      0.106041    8.911
LOG[SIZE]     2.14319       0.109983    19.486
PMATXLOG[SIZE] 0.0167201    0.0129698   1.289
PRECXLOG[SIZE] 0.0279765    0.0109755   2.549
RESLXLOG[SIZE] 0.0320644    0.0146620   2.187
FLEXXLOG[SIZE] 0.00773647   0.0165750   0.467
TEAMXLOG[SIZE] 0.00948770   0.0184777   0.513
log[ACAP]     1.03455      0.308768    3.351
log[AEXP]     0.153244     0.356819    0.429
log[CPLX]     0.970973     0.222219    4.369
log[DATA]     0.794825     0.231921    3.427
log[DOCU]     1.84466      0.551029    3.348
log[LTEX]     0.130685     0.441287    0.296
log[PCAP]     1.31465      0.333095    3.947
log[PCON]     0.632973     0.372297    1.700
log[PEXP]     0.904631     0.443070    2.042
log[PVOL]     1.24179      0.288457    4.305
log[RELY]     0.923364     0.346244    2.667
log[RUSE]     -0.702669    0.608173    -1.155
log[SCED]     1.20785      0.281471    4.291
log[SITE]     0.713771     0.531717    1.342
log[STOR]     0.916076     0.374280    2.448
log[TIME]     1.50943      0.367092    4.112
log[TOOL]     0.967892     0.358490    2.700

R Squared:           0.963891
Sigma hat:           0.317665
Number of cases:     161
Degrees of freedom:  137

Summary Analysis of Variance Table
Source      df      SS      MS      F      p-value
Regression  23    369.036  16.045  159.00  0.0000
Residual    137   13.8249  0.100911

```

Table 6.8. Fit of 161 project data (Three-dimensional TOOL)

Table 6.9 shows the standard error and bias estimates obtained from the two bootstrap objects. The standard error estimates compare with the usual estimates in Table

6.7 and Table 6.8, obtained from the usual estimates with the two datasets, respectively. Note that the usual nominal regression estimates of standard error are the ideal bootstrap estimates of standard error as the number of boots $\rightarrow \infty$ adjusted by $([n / (n - k)]^{1/2})$. The bootstrap estimates of bias can be calculated as shown in (EQ 7) from the idea “ $\hat{\theta}^*$ is to $\hat{\theta}$ as $\hat{\theta}$ is to θ ”. Those bootstrap estimates of bias for all predictor variables in the regression are relatively small because the usual normal regression estimates are unbiased.

(EQ 7)

$$\text{Bias}_{\text{bootstrap}} = \hat{\theta}_i^*(.) - \hat{\theta}_i$$

Where

$\hat{\theta}_i$: i^{th} coefficient in the regression

$\hat{\theta}_i^*(.)$: the average of $\hat{\theta}_i^*$'s

When compared with the bootstrap estimates of standard error and bias for log[TOOL] obtained from the one-dimensional TOOL dataset, the bootstrap estimates of standard error are a little bit smaller. Also, the bias estimate is reasonably small. That is, the normal regression model with three-dimensional TOOL ratings is better fit to the observations in the second dataset. Again, the differences are not great, but the results are stable and in the right direction.

Coefficients	One-Dimensional TOOL		Three Dimensional TOOL	
	Std-Error	Bias	Std-Error	Bias
Intercept	0.129501	-0.007315	0.124337	-0.006441
LOG[SIZE]	0.129664	0.024798	0.125930	0.027982
PMATXLOG[SIZE]	0.016358	-0.002399	0.016306	-0.003945
PRECXLOG[SIZE]	0.014236	-0.001885	0.014023	-0.001870
RESLXLOG[SIZE]	0.016678	0.000046	0.016437	0.001374
FLEXXLOG[SIZE]	0.018616	-0.000723	0.018495	-0.000424
TEAMXLOG[SIZE]	0.020669	0.001350	0.020203	0.000824
log[ACAP]	0.300547	-0.015081	0.306939	-0.015901
log[AEXP]	0.458397	0.099686	0.408997	0.106427
log[CPLX]	0.214933	-0.026814	0.206705	-0.032947
log[DATA]	0.261370	-0.015302	0.251062	0.010646
log[DOCU]	0.672710	-0.080994	0.677146	-0.034751
log[LTEX]	0.553945	0.056942	0.545734	0.078652
log[PCAP]	0.352869	-0.019446	0.351389	-0.016883
log[PCON]	0.451627	0.010159	0.455042	0.011140
log[PEXP]	0.480671	-0.126271	0.456292	-0.136301
log[PVOL]	0.308297	0.006236	0.318286	-0.000514
log[RELY]	0.402958	-0.011795	0.409612	-0.039123
log[RUSE]	0.831980	-0.052474	0.810184	-0.052127
log[SCED]	0.348284	0.015908	0.331593	-0.024937
log[SITE]	0.584291	0.086295	0.589103	0.114096
log[STOR]	0.969903	0.382749	0.967008	0.438698
log[TIME]	0.604294	-0.155874	0.585690	-0.192345
log[TOOL]	0.392941	0.030489	0.359567	0.042730

Table 6.9. Bootstrap standard-error and bias

Three confidence intervals for each coefficient estimate such as normal theory confidence interval, bootstrap confidence intervals using both percentile method and BCa method are calculated in Arc by typing the following command.

```
> (send c :display-confidence-intervals)
```

For the simplicity of the comparison, Table 6.10 shows the percentile bootstrap confidence intervals for the coefficient estimates that use percentiles of the empirical distribution of the bootstrap estimates to estimate percentiles of the true distribution of the

coefficients. The detailed information on the three confidence intervals are available in (Iain Pardoe, 2000). The default level for the confidence intervals is 95%. In repeated datasets, the true population mean will be included in the 95% confidence intervals. When compared with the normal regression estimates for the coefficients in Table 6.7 and Table 6.8, respectively, the percentile bootstrap confidence intervals indicates that the resampling results agree closely with those obtained from standard methods.

Coefficients	One-Dimensional TOOL	Three Dimensional TOOL
Intercept	(0.693841 1.18144)	((0.701122 1.20092)
LOG[SIZE]	(1.90445 2.41717)	(1.92887 2.42048)
PMATXLOG[SIZE]	(-0.0208585 0.0447204)	(-0.0211637 0.0422511)
PRECXLOG[SIZE]	(-0.00222309 0.0533572)	(-0.00235009 0.0523237)
RESLXLOG[SIZE]	(-0.0005743 0.0641629)	(0.0016363 0.0655087)
FLEXXLOG[SIZE]	(-0.0275789 0.0452344)	(-0.0277733 0.0464793)
TEAMXLOG[SIZE]	(-0.0303952 0.0515796)	(-0.0296432 0.0478585)
log[ACAP]	(0.434194 1.67577)	(0.449679 1.64775)
log[AEXP]	(-0.689046 1.11136)	(-0.505828 1.0968)
log[CPLX]	(0.514885 1.34717)	(0.530155 1.35197)
log[DATA]	(0.269012 1.25981)	(0.310233 1.29736)
log[DOCU]	(0.490531 3.14065)	(0.479138 3.08239)
log[LTEX]	(-0.863201 1.38017)	(-0.874451 1.30694)
log[PCAP]	(0.618606 1.98188)	(0.585507 1.99207)
log[PCON]	(-0.212051 1.59456)	(-0.211628 1.54548)
log[PEXP]	(-0.0449577 1.82346)	(-0.166268 1.70002)
log[PVOL]	(0.647702 1.83542)	(0.634722 1.91595)
log[RELY]	(0.00599703 1.59866)	(-0.117272 1.57073)
log[RUSE]	(-2.32187 0.900462)	(-2.21625 0.889567)
log[SCED]	(0.592632 1.99417)	(0.607387 1.9011)
log[SITE]	(-0.327599 1.92736)	(-0.248572 2.02885)
log[STOR]	(0.048242 3.24556)	(0.0102692 3.18507)
log[TIME]	(0.194714 2.47158)	(0.173605 2.36188)
log[TOOL]	(0.184884 1.67873)	(0.340032 1.71783)

Table 6.10. Percentile bootstrap confidence intervals

Once again, most of the percentile confidence intervals from three dimensional TOOL data set are narrower than those obtained from one-dimensional dataset.

Especially, the confidence interval for the coefficient of $\log[\text{TOOL}]$ from three-dimensional dataset has a smaller region than that from one-dimensional dataset shown in Figure 6.3. That is, the extended three dimensional TOOL rating scales (TCOV, TINT, and TMAT) gives a better fit of regression estimates than the one-dimensional TOOL rating scale does.

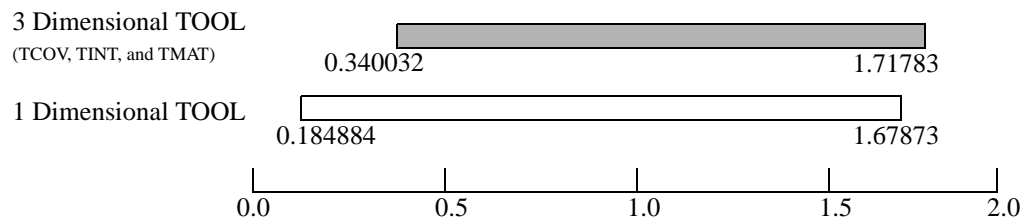


Figure 6.3. Percentile bootstrap confidence intervals for $\log[\text{TOOL}]$

6.3 Summary of Cross Validation

The two cross validation methods (cross-validation by splitting data and bootstrap supported by Arc) and their simulation results are described in the above sections. In the cross validation by splitting data, two experiments are performed with 15 intentionally selected validation subsamples and 46 randomly selected validation subsamples. In the first experiment, the cross-validation result indicates that the observations in the validation dataset using three dimensional TOOL rating scale is better fit to the normal regression obtained from the same construction dataset than those in the validation set using the one dimensional TOOL rating scale. In the second experiment with the 46 randomly selected validation dataset, the cross-validation result also shows that PRESS for the dataset using the three dimensional TOOL rating scales is smaller than that for the dataset using one

dimensional TOOL rating scale. That indicates that better regression models can be found when the extended TOOL rating scales (TCOV, TINT, and TMAP) are used.

In the bootstrap simulations, the histograms and the probability plots for the coefficient estimates with 1000 boots indicates that there is near-normality for both one dimensional and three dimensional TOOL datasets. With comparison with standard error and bias obtained from the dataset using one dimensional TOOL rating scale, the simulation result indicates that those values can be reduced when the extended TOOL rating scales are used. The percentile bootstrap confidence intervals also shows that better regression models can be found with the dataset using three dimensional TOOL ratings.

Since only 15 out of 161 project data have information about the three TOOL rating scales, the simulation results for the cross-validation show a little bit of improvement in the values such as PRESS, standard error, and so on. If the TOOL rating for every observation is determined by the extended three TOOL rating scales, there may show more improvements in those values.

7.0 Conclusions

This chapter summarizes the key contributions of this dissertation and suggests some research directions to be pursued in the future.

7.1 Summary of Contributions

1. The classification of CASE tools based on the breadth of their coverage in a software life-cycle is suggested to provide a clearer conceptual basis than any other classifications. It gives a visual coverage of CASE tools and technologies horizontally and vertically in a software production process.
2. As an extension of COCOMO II, a set of three dimensional TOOL rating scales is proposed to effectively evaluate CASE tools. In the Bayesian result of COCOMO II.2000, the productivity range of the effort multiplier TOOL shows a relatively high pay-off in software productivity improvement activities. However, the TOOL ratings focuses on only functionalities supported by CASE tools without considering other important factors such as degree of tool integration, tool maturity, and user support. Thus, the one dimensional TOOL rating scale is extended into the three dimensional rating scales (TCOV, TINT, TMAT) to reflect those factors.
3. In order to find the best fit of the weighting values for the extended three dimensional TOOL rating scales, the Bayesian approach is used to combine Delphi (expert-judged) and Sampling (data-determined) information. When the TOOL rating is determined by the sum of the products of each weighting value obtained from Bayesian and

rating, it yields a prediction accuracy of $PRED(.10) = 87\%$ on 15 projects. Whereas, the COCOMO II Bayesian that use the one-dimensional TOOL rating scale yields a poorer accuracy of $PRED(.10) = 67\%$.

4. The research model demonstrates a method to calibrate an individual contribution to a set of multidimensional parameter contributions by breaking one TOOL rating scale into three TCOV, TINT and TMAT. This method can applied to the other parameters in the COCOMO II model and also in other parametric models.

5. It is very hard to collect new project data in practice. The two cross validation methodologies (Data Splitting and Bootstrap) are introduced to validate the research model. The validation set in Data Splitting and the boots in Bootstrap are used for the validation of a regression model in the same way as in the newly collected dataset. The simulation results indicates that better regression models can be found when the extended TOOL rating scales are used.

7.2 Future Research Directions

1. A huge number of CASE tools are produced every single day and new CASE technologies are introduced by researchers in the field. Thus, the extended TOOL rating scales and the classification should be refined with more available CASE tools and technologies in the future CASE market. The behavioral analysis of CASE tool effects should be refined, too.

2. For this research, only a small set of project data, 15 projects, that have information about the extended three dimensional TOOL rating scales are adopted from the COCOMO II database. Once enough project data with three dimensional TOOL rating scales is collected, the weighting values for the three TOOL rating scales should be recalibrated via the Bayesian approach.

3. Data for interaction among the three TOOL rating scales should be collected to assess the individual impact as well as the combined impact on software development effort.

4. The calibration for the next version of COCOMO II should be performed with and without the extended three dimensional TOOL rating scales. The analyzed results of the calibration will be compared based on the prediction accuracy.

5. The Improvement of CASE tools to get a higher productivity involves many economic problems such as software acquisition, training, maintenance costs, and so on. TOOL ROI (Return On Investment) should be analyzed to determine under what conditions the investment on CASE tools can be economically justified to improve software development productivity. It would be very useful to support ROI analysis via a software tool and integrate it with USC-COCOMO II software.

8.0 Bibliography

Alan M. Christie (1994), A Practical Guide to the Technology and Adoption of software Process Automation, CMU/SEI-94-007, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Alfonso Fuggetta (1993, December), Politecnico di Milano and CEFRIEL, A Classification of CASE Technology, IEEE Computer.

Banker R, Robert J. Kauffman, & Rachna Kumar (1992), An Empirical Test of Object-based Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment.

Banker R., Chang, & C. Kemerer (1994), Evidence on Economies of Scale in Software Development, Information and Software Technology.

Bill Curtis (2000, July/August), Building Accelerated Organizations, IEEE Software, 72-74.

Boehm, B. W., Software Engineering Economics (1981), Prentice Hall, New Jersey.

Boehm, B. W. (1984, January), Software Engineering Economics, IEEE Transactions on Software Engineering, 4-21.

Boehm, B. W. & Walker Royce (1989), Ada COCOMO and Ada Process Model, Proceedings of 5th COCOMO User's Group Meeting, Software Engineering Institute, Pittsburgh, Pennsylvania.

Boehm, B. W. et al. (2000), Software Cost Estimation with COCOMO II, Prentice Hall, New Jersey.

Bradford Clark (1997), The Effects of Software Process Maturity on Software Development Effort, Ph.D dissertation, University of Southern California, Los Angeles, California.

Bradford Clark, Sunita Chulani, & Boehm, B. W. (1998), Calibration Results of COCOMO II.1997, International Conference on Software Engineering, Japan.

Bradley Efron & Robert J. Tibshirani, (1993), An Introduction to the bootstrap, Chapman & Hall, 1993.

Caper Jones (1996), Applied Software Metrics, McGraw Hill.

Caper Jones (1998), Estimating Software Cost, McGraw Hill.

Conte, S., H. Dunsmore, V. Shen (1986), Software Engineering Metrics and Models, Benjamin/Cummings, Menlo Park, California.

Dacs (1998), Modern Empirical Cost and Schedule Estimation Tools, <http://www.dacs.com/techs/estimation/comparison.shtml>.

David Sharon & Rodney Bell (1995, March), TOOLS THAT BIND: Creating Integrated Environments, IEEE Software, 76-85.

Dennis Cook and Sanford Weisberg (1999), Applied Regression Including Computing and Graphics, Wiley Series.

D. Reifer, P. Kane, & D. Willens (1989), SoftCost-R User Guide, Reifer Consultants, Inc., Torrance, California.

D. Reifer, P. Kane, & T. Pillsbury (1989), SoftCost-Ada User Guide, Reifer Consultants, Inc., Torrance, California.

Edward E. Leamer (1978), Specification Searches: Ad hoc Inference with Nonexperimental Data, Wiley Series.

Elliot J. Chikofsky & Burt L. Rubenstein (1988, March), CASE: Reliability Engineering for Information Systems, IEEE Software, 11-16.

Forte, G. and McCully, K. (1991), CASE Outlook: Guide to Products and Services, CASE Consulting Group, Lake Oswego, Oregon.

Gerge G. Judge et al. (1985), The Theory and Practice of Econometrics, 2nd Edition, John Wiley and Sons.

Iain Pardoe (2000, February), An Introduction to Bootstrap Methods using Arc, Technical Report 631, University of Minnesota, St. Paul, Minnesota.

Ian Sommerville (1995), Software Engineering, 5th Edition, Addison-Wesley Pub Co..

IFPUG, (1994), Function Point Counting Practices Manual, Release 4.0, International Function Point Users Group (IFPUG).

John Neter et. al., Applied Linear Regression Models, 3rd Edition, IRWIN, 1996.

Lawrence H. Putnam and Ware Myers (1992), Measures for Excellence, Yourdon Press Computing Series.

Michael R. Chernick, Bootstrap Methods: A Practitioner's Guide, John Wiley & Sons, Inc..

Ovum (1995, June), Contents and Services, Ovum Ltd., UK.

Parametric (1995), Parametric Cost Estimating Handbook, Naval Sea Systems Command.

Paulk, M. C. et al. (1993), Capability Maturity Model for Software, Version 1.1, CMU/SEI-TR-25. Carnegie Mellon University, Pittsburgh, Pennsylvania.

Peter Lempp and Rudolf Lauber (1988, June), What Productivity Increases to Expect from CASE Environment: Result of User Survey, Productivity: *Progress, Prospects, and Payoff*, 13-19.

Phillip I. Good (1999), Resampling Methods: A Practical Guide to Data Analysis, Birkhauser.

PRICE (1998), PRICE Software Estimating Model, <http://www.pricesystems.com>, PRICE Systems, L L C.

Randall W. Jensen, A Comparison of the Jensen and COCOMO Schedule and Cost Estimation Model, Proceedings of the International Society of Parametric Analysis, 96-106.

Robert C. Tausworthe (1981, April), Deep Space Network Software Cost Estimation Model, Jet Propulsion Laboratory Publication 81-7, Pasadena, California.

Robert E. Park (1988, November), The Central Equation of the PRICE Software Cost Model, 4th COCOMO User's Group meeting.

Robert E. Park (1992), Software Size Measurement: A Framework for Counting Source Statements, CMU/SEI-92-TR-20, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Robert Firth et al. (1987), A Guide to the Classification and Assessment of Software Engineering Tools, CMU/SEI-87-TR-10, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Roger S. Pressman (1992), Software Engineering: A Practitioner's Approach, 3rd Edition, McGraw-Hill, Inc..

Sanford Weisberg (1999, December), Cross-Validation in Arc, <http://www.stat.umn.edu/arc/crossvalidation/crossvalidation/crossvalidation.html>.

STSC (1995, April), Report on Project Management and Software Cost Estimation Technologies, STSC-TR-012, System Technology Support Center.

Sunita Chulani, Boehm, B. W., & Bert Steece (1999, July/August), Bayesian Analysis of Empirical Software Engineering Cost Models, IEEE Transactions on Software Engineering, 513-583.

Sunita Chulani & Boehm, B. W., Modeling Software Defect Introduction Removal: COQUALMO (CONstructive QUALity MOdel), USC-CSE-99-510, The Center for Software Engineering, University of Southern California, Los Angeles, California.

Terry Bollinger (2000, July/August), Building Tech-Savvy Organizations, IEEE Software, 73-75.

USC-CSE (1999), COCOMO II and COQUALMO Data Collection Questionnaire, <http://sunset.usc.edu/COCOMOII/cocomo.html>, The Center for Software Engineering, University of Southern California, Los Angeles, California.

Vicky Mosley (1992, May), How to Assess Tools Efficiently and Quantitatively, IEEE Software.

Wasserman, A. I. (1990), Tool integration in software engineering environments., In Proc. Int. Workshop on Environments, Berlin, 137-149.

Weisberg, S. (1985), Applied Linear Regression, John Wiley & Sons, New York.

W.E. Griffiths, R.C. Hill, and G.G. Judge (1993), Learning and Practicing Econometrics, John Wiley & Sons, Inc., New York.

William W. Hines and Douglas C. Montgomery (1972), Probability and Statistics In Engineering and Management Science, Wiley Series.

[Yourdon 1996] Ed Yourdon, *“Death March, Managing Mission Impossible projects”*, Prentice Hall.

Appendix A. Delphi Questionnaire

Delphi Questionnaire of COCOMO II TOOL Components

Objective: Obtain consensus-based weighted value for each of the extended three TOOL rating scale parameters to initiate Bayesian analysis.

Coordinator:

Jongmoon Baik
Graduate Research Assistant
USC-CSE
jobaik@sunset.usc.edu
Phone: 1-213-740-6470
FAX: 1-213-740-6470

Participants:

COCOMO II team at USC and interest Affiliates

Approach:

Coordinator provides participants with Delphi Instrument. Participants input weighted values for three TOOL rating scales matching their experience. They provide the response to the coordinator. The results obtained by Delphi technique will be provided to the participants.

Delphi Instrument

As one of extensions to COCOMO II model, we developed three dimensional TOOL rating scales such as *completeness of tool coverage, degree of tool integration, and tool maturity/user support* to assess the more accurate impact of software tools on predicting software development efforts. we plan to do a Bayesian analysis for calibrating the weighted values of three new TOOL rating scales. This Bayesian analysis requires information on the relative dispersion of the expert-judgement-based a-priori weighted value for each of the extended three TOOL rating scales.

We plan to do this Delphi exercise involving the COCOMO II affiliates. Attached is the delphi questionnaire on which you enter your best estimate of the initial weighted values based on your valuable experience. The result of the Delphi will determine the a-priori values and dispersions for the new three TOOL rating scale calibration. We estimate that this will take you less than 10 minutes to complete.

The following is the initial experimental result of the weighted values for the three TOOL rating scale we found by using Bayesian analysis. You can input just those values based on your experience.

	TCOV	TINT	TMAT	TOTAL
Weighted Values	0.444	0.349	0.207	1.00

Delphi Data Collection Instrument

Participant's Information:

Name		E-mail address	
Phone No.		Fax No.	
Snail Mail			

USC-CSE Contact Point:

For any questions or comments, please contact:
Jongmoon Baik
Graduate Research Assistant
USC-CSE
jobaik@sunset.usc.edu
1-213-740-6505

Data Submission Address:

Jongmoon Baik
Center for Software Engineering
Department of Computer Science
Salvatori 329
University of Southern California
941 W. 37th Place
Los Angeles, CA 90089-0781

COCOMO II TOOL Components

1. Completeness of Activity Coverage (TCOV)

This measure captures how well the software tools support given tasks in a software life-cycle process based on functionality and breadth of life-cycle coverage. It ranges from simple tools, Very Low, to Groupware and Distributed systems, Extra High.

	Very Low	Low	Nominal	High	Very High	Extra High
Completeness of Activity Coverage	Text-Based Editor, Basic 3GL Compiler, Basic library Aids, Basic Text-based Debugger, Basic Linker	Graphical Interactive Editor, Simple Design Language, Simple Programming Support Library, Simple Metrics/Analysis Tool	Local Syntax Checking Editor, Standard Template Support Document Generator, Simple Design Tools, Simple Stand-alone Configuration Management Tool, Standard Data Transformation Tool, Standard Support Metrics Aids with Repository, Simple Repository, Basic Test Case Analyzer	Local Semantics Checking Editor, Automatic Document Generator, Requirement Specification Aids and Analyzer, Extended Design Tools, Automatic Code Generator from Detailed Design, Centralized Configuration Management Tool, Process Management Aids, Partially Associative Repository (Simple Data Model Support), Test Case Analyzer with Spec. Verification Aids, Basic Reengineering & Reverse Engineering Tool	Global Semantics Checking Editor, Tailorable Automatic Document Generator, Requirement Specification Aids and Analyzer with Tracking Capability, Extended Design Tools with Model Verifier, Code Generator with Basic Round-Trip Capability, Extended Static Analysis Tool, Basic Associative, Active Repository (Complex Data Model Support), Heterogeneous N/W Support Distributed Configuration Management Tool, Test Case Analyzer with Testing Process Manager, Oracle Support, Extended Reengineering & Reverse Engineering Tools	GroupWare systems, Distributed Asynchronous Requirement Negotiation and Trade-off tools, Code Generator with Extended Round-Trip Capability, Extended Associative, Active Repository, Spec-based Static and Dynamic Analyzers, Pro-active Project decision Assistance

2. Degree of Tool Integration (TINT)

This is the measure of how well integrated software tools in development environments. If software tools are highly independent of each other, TINT is Very Low. If software tools are very tightly integrated, TINT is Extra High.

	Very Low	Low	Nominal	High	Very High	Extra High
Degree of Tool Integration	Individual File Formats for Tools (No Conversion Aid), No Activation Control for Other Tools, Different User Interface for each Tools, Fundamental Incompatibilities among Process Assumptions and Object Semantics	Various File Formats for Each Tools (File Conversion Aids), Message Broadcasting to Tools, Some Standardized User Interfaces among Tools, difficult Incompatibilities among Process Assumptions and Object Semantics	Shared-Standard Data Structure, Message Broadcasting through Message Server, Standard User Interface Use among Tools, Reasonably Workable Incompatibilities among Process Assumptions and Object Semantics	Shared Repository, Point-to-Point Message Passing, Customizable User Interface Support, Largely Workable Incompatibilities among Process Assumptions and Object Semantics	Highly Associative Repository, Point-to-Point Message Passing Using reference for Parameters, Some level of Different User Interface, Largely Consistent among Process Assumptions and Object Semantics	Distributed-Associative Repository, Extended Point-to-Point Message Passing for Tool Activation, Complete Set of User Interface for different level of Users, Fully Consistent among Process Assumptions and Object Semantics

3. Tool Maturity and User Support (TMAT)

This rating is the measure of how mature software tools in the market and what kind of services are provided by software vendors. The ratings are defined in terms of the software tools' equivalent level of the survival length after their release and services. For tool maturity, a very low rating is for the software tools whose version is in pre-release beta-test process. A extra high rating is fro the software tools which is available more than 3 years after their release. For user support, a very low rating is for the software tools that is supplied with only simple documents. A extra high rating is for the software tools that is supplied with strong on-site support group from the vendors. Some amount of judgement is required to combine two aspects of tool maturity and user supports.

	Very Low	Low	Nominal	High	Very High	Extra High
Tool Maturity and User Support	Version in pre-release beta-test, Simple documentation and help	Version on market/available less than 6 month, Updated documentation, help available	Version on market/available between 6 months and 1 year, On-line help, tutorial available	Version on market/available between 1 and 2 years, On-line User Support Group	Version on market/available between 2 and 3 years, On-Site Technical User Support Group	Version on market/available more than 3 years

4. Weighted Values for the COCOMO II TOOL Components

The COCOMO II TOOL rating can be determined by weighted sum of the three TOOL rating scales as the following.

$$TOOL = b_1 \cdot TCOV + b_2 \cdot TINT + b_3 \cdot TMAT$$

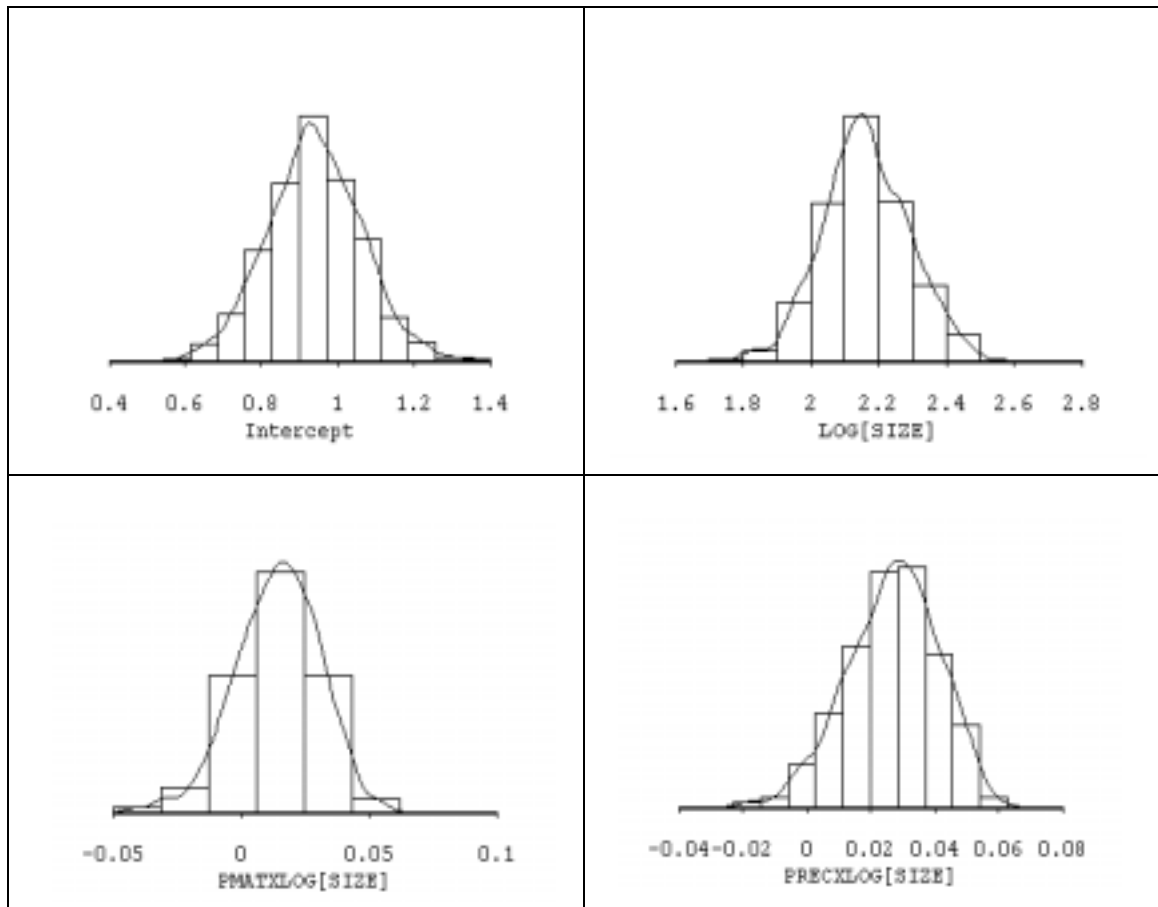
$$\sum_{i=1}^3 b_i = 1$$

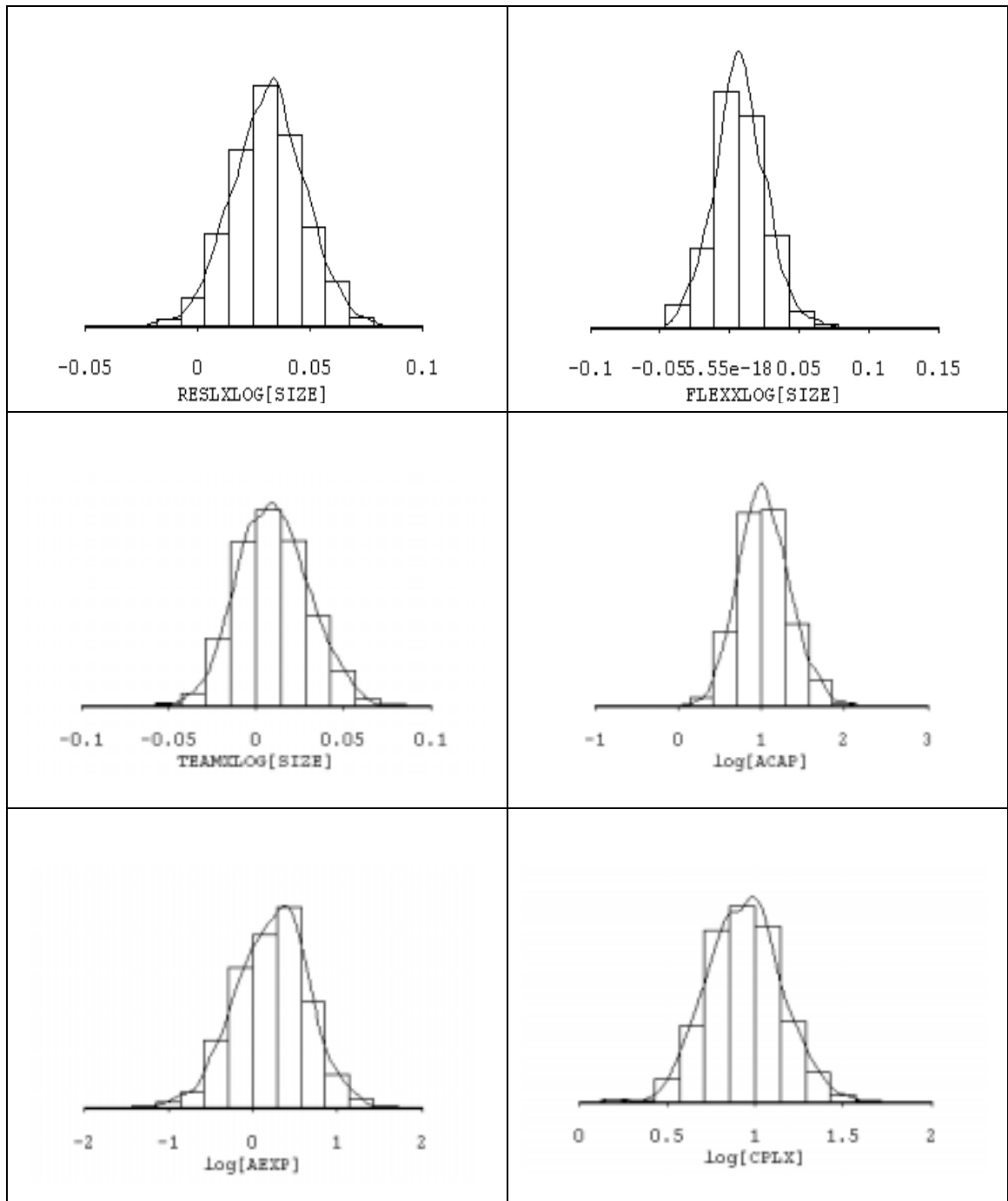
Please, input the relative weighted values for the extended TOOL rating scales based on your experience.

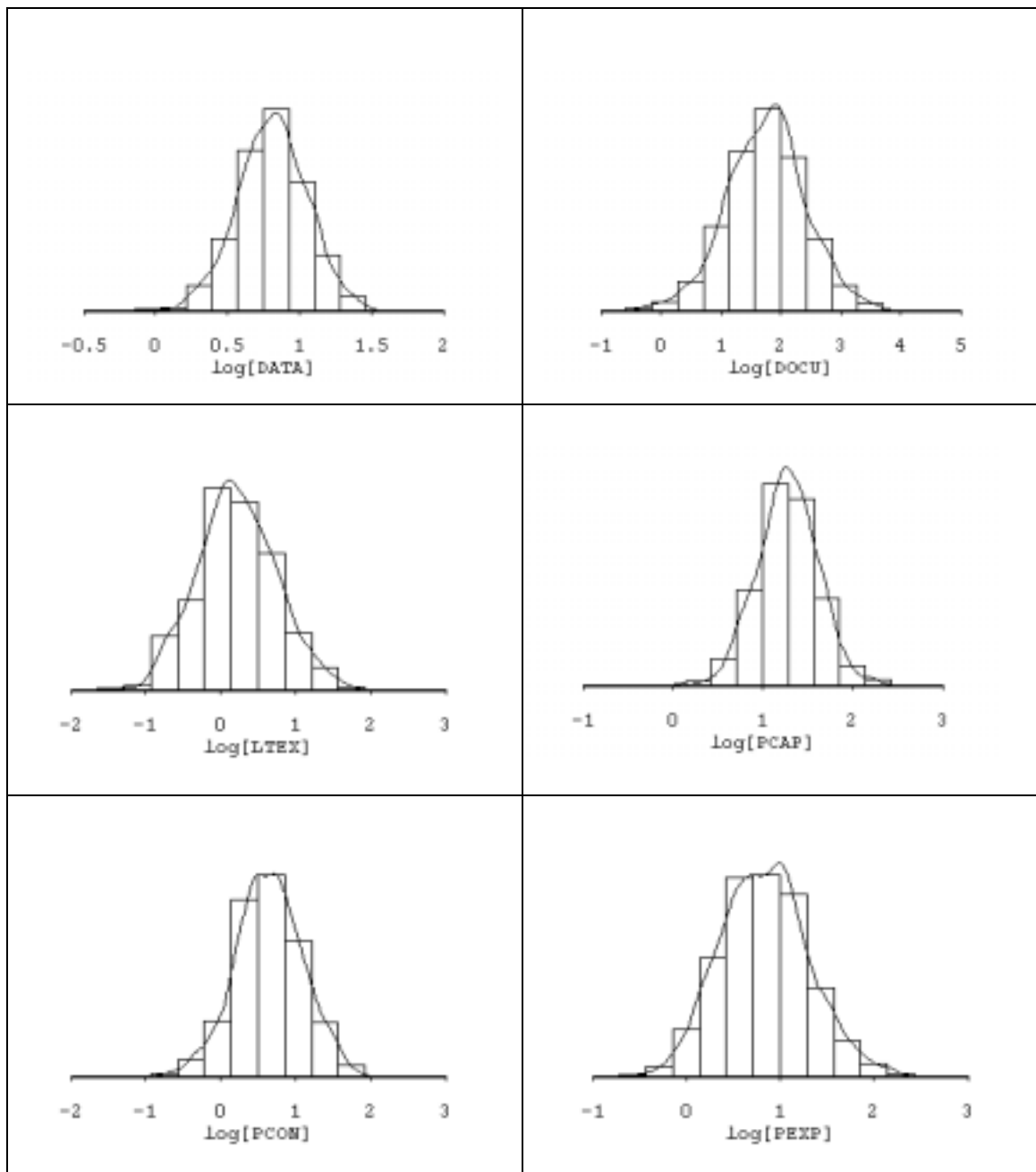
	TCOV	TINT	TMAT	TOTAL
Weighted Values				1.00

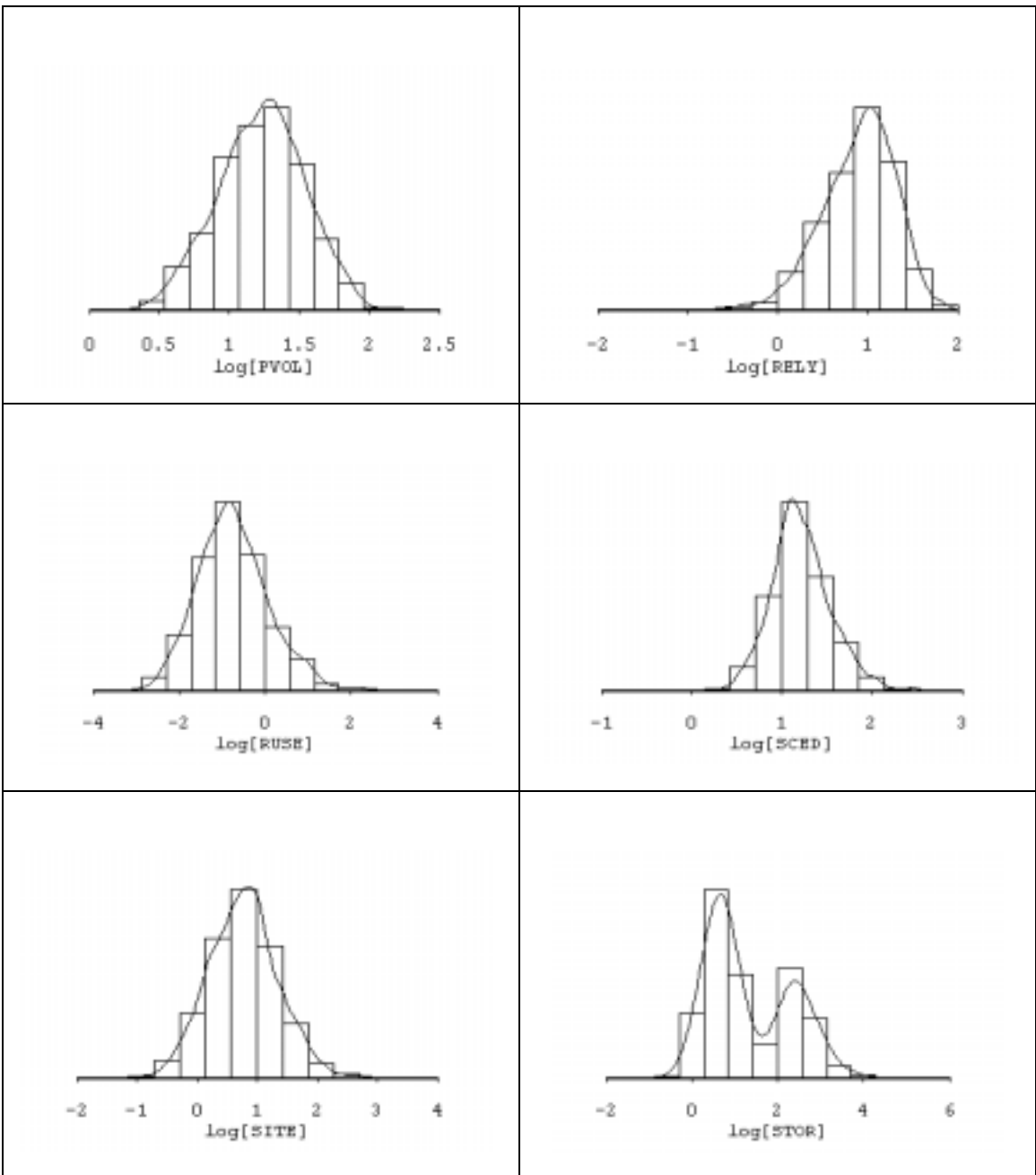
Appendix B: Bootstrap Histograms and Probability Plots

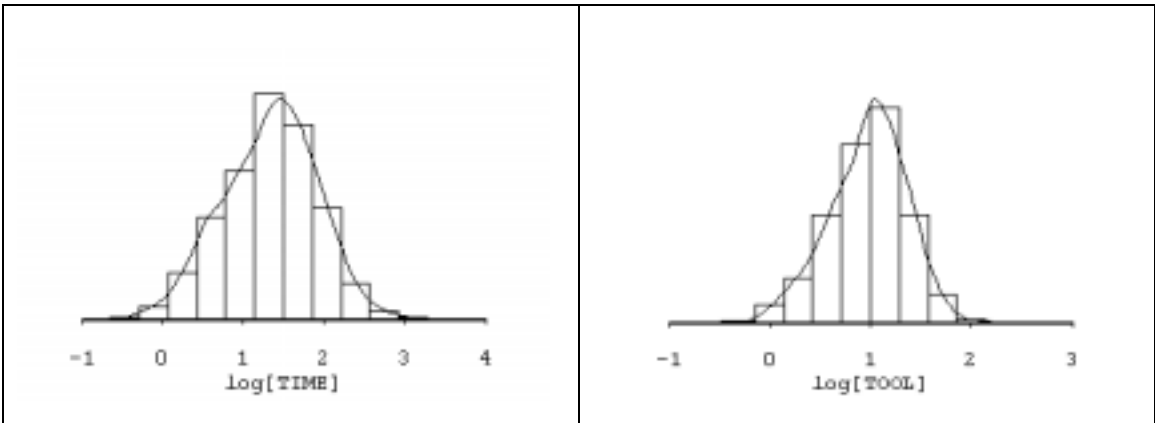
B.1 Histograms (One dimensional TOOL)



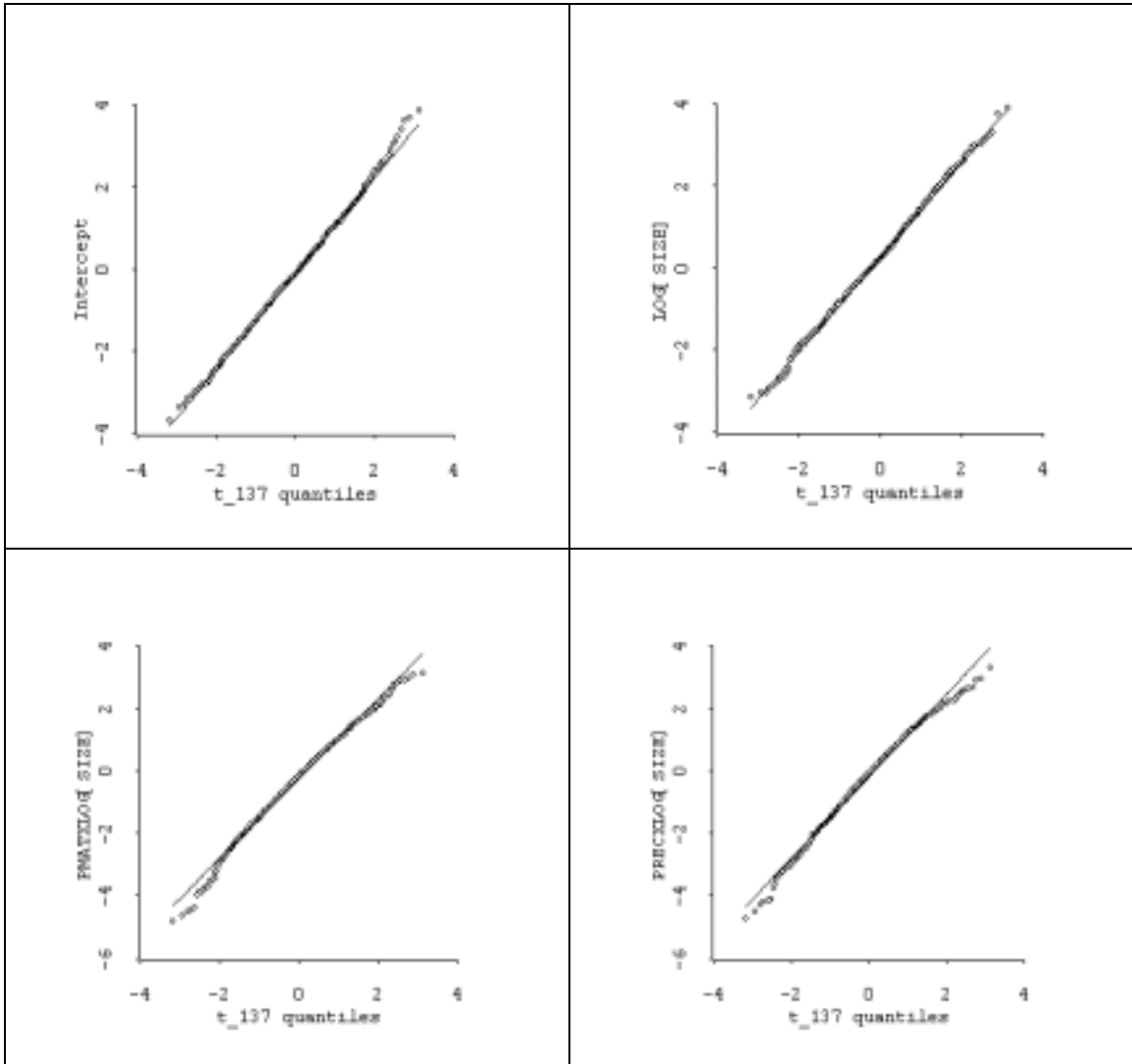


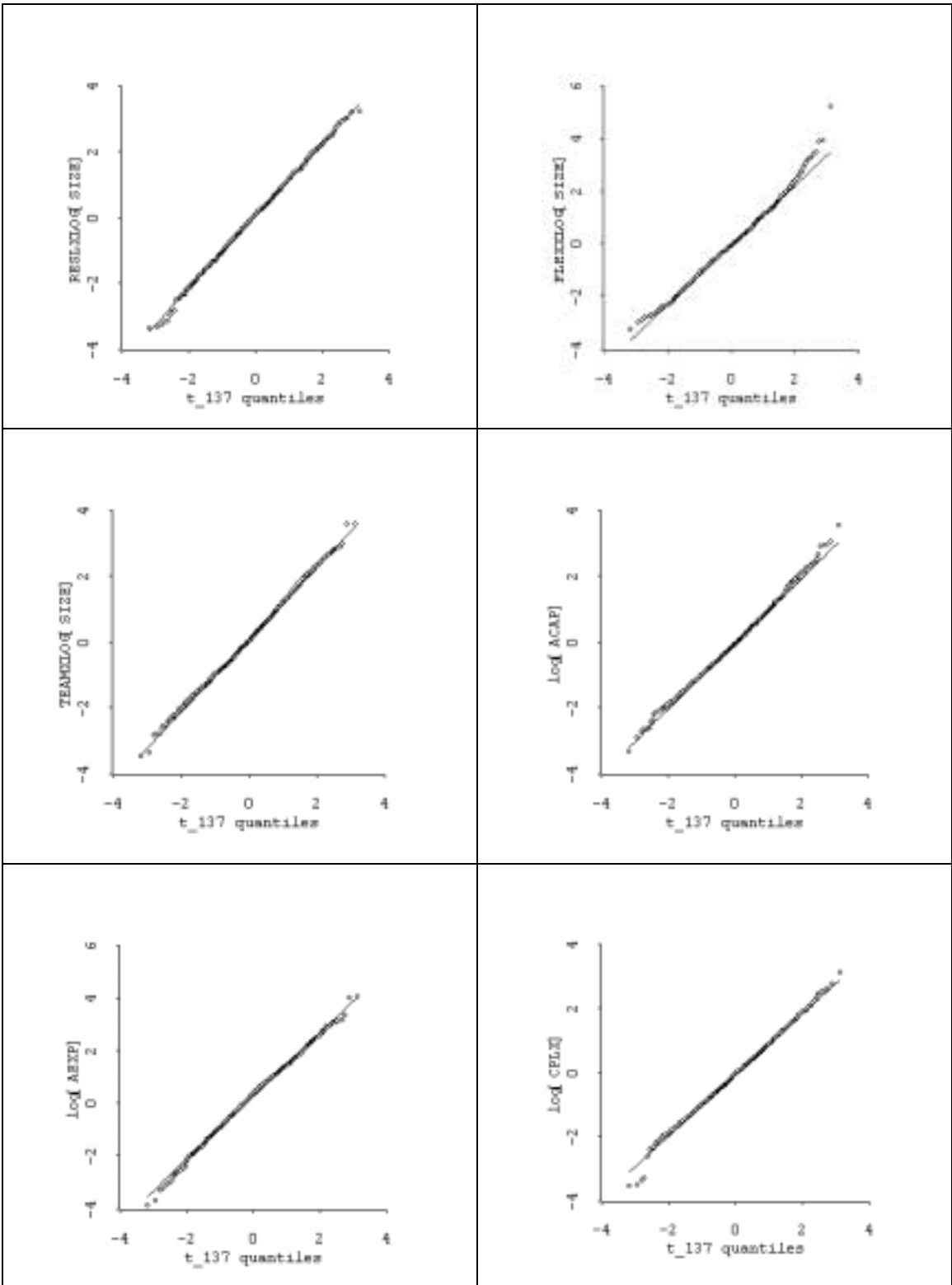


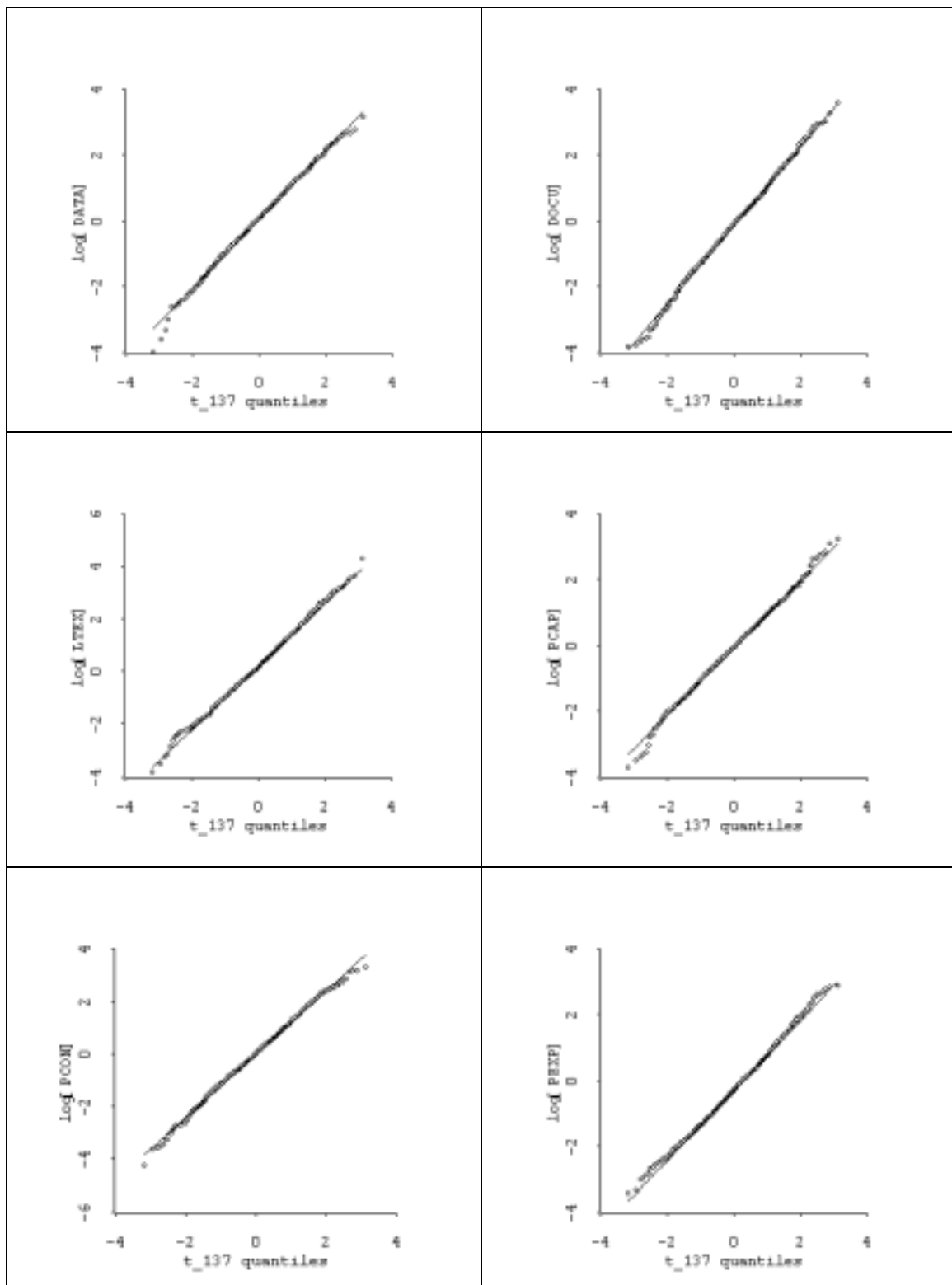


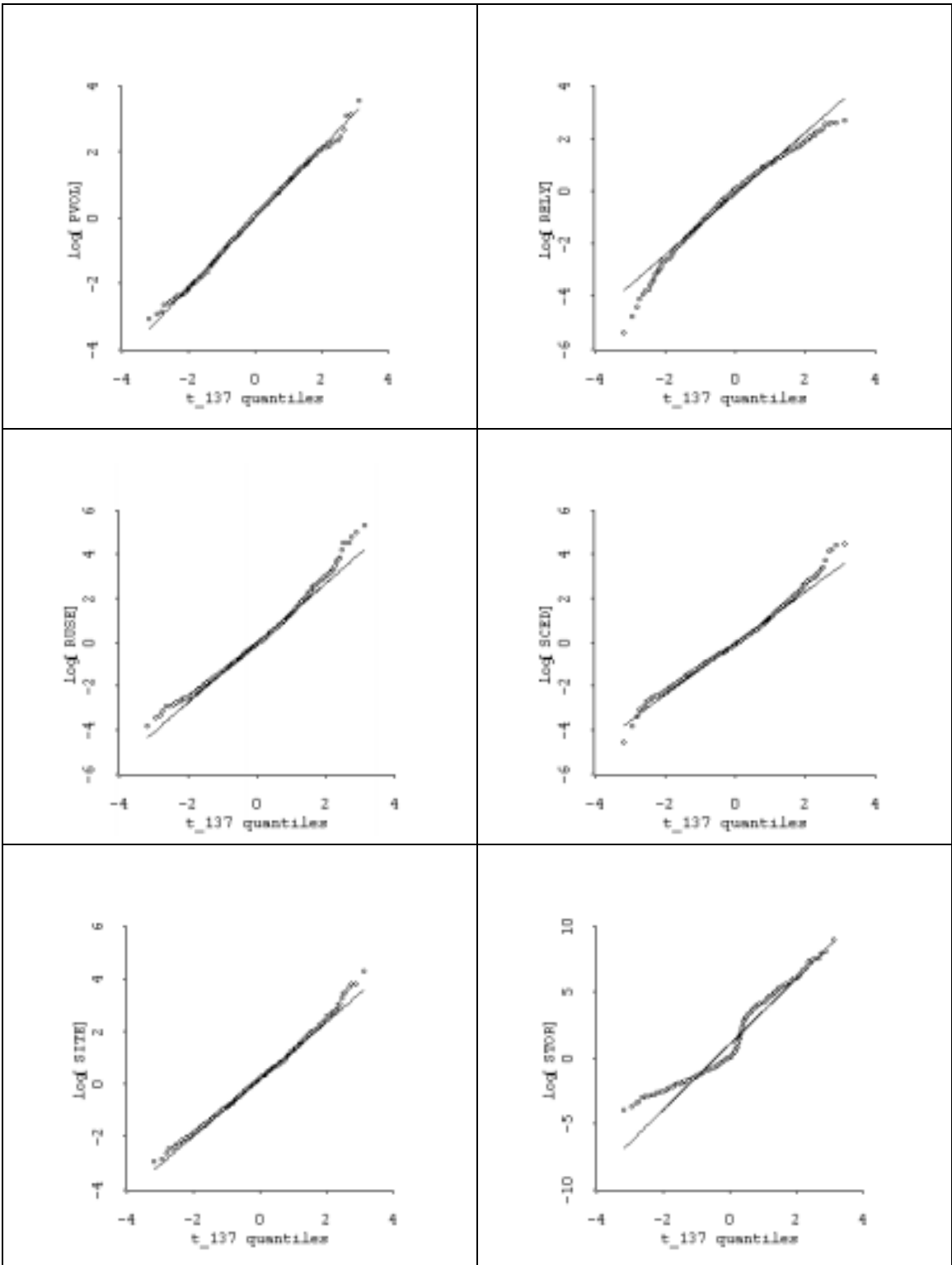


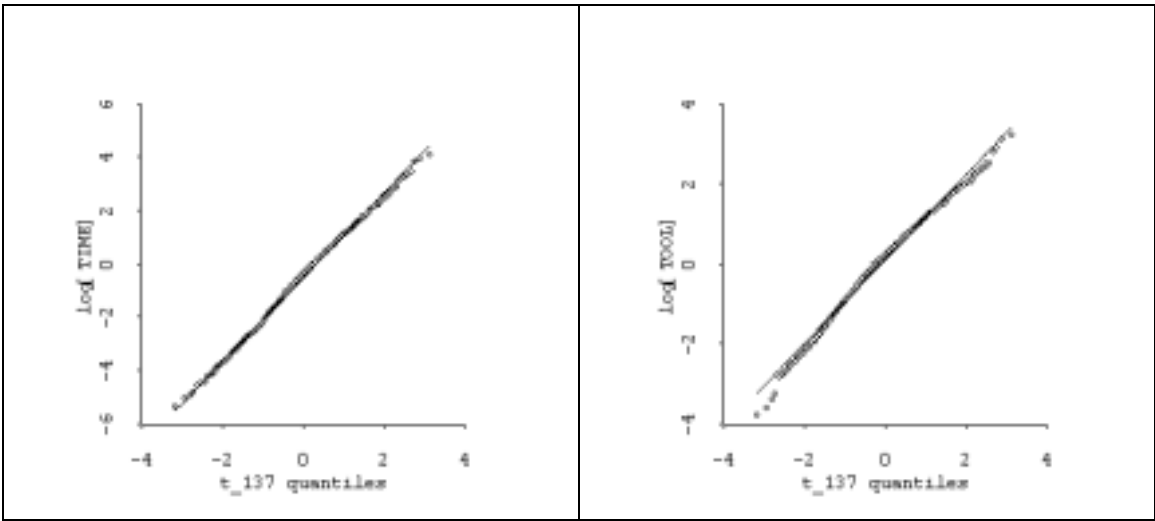
B.2 Probability Plots (One dimensional TOOL)



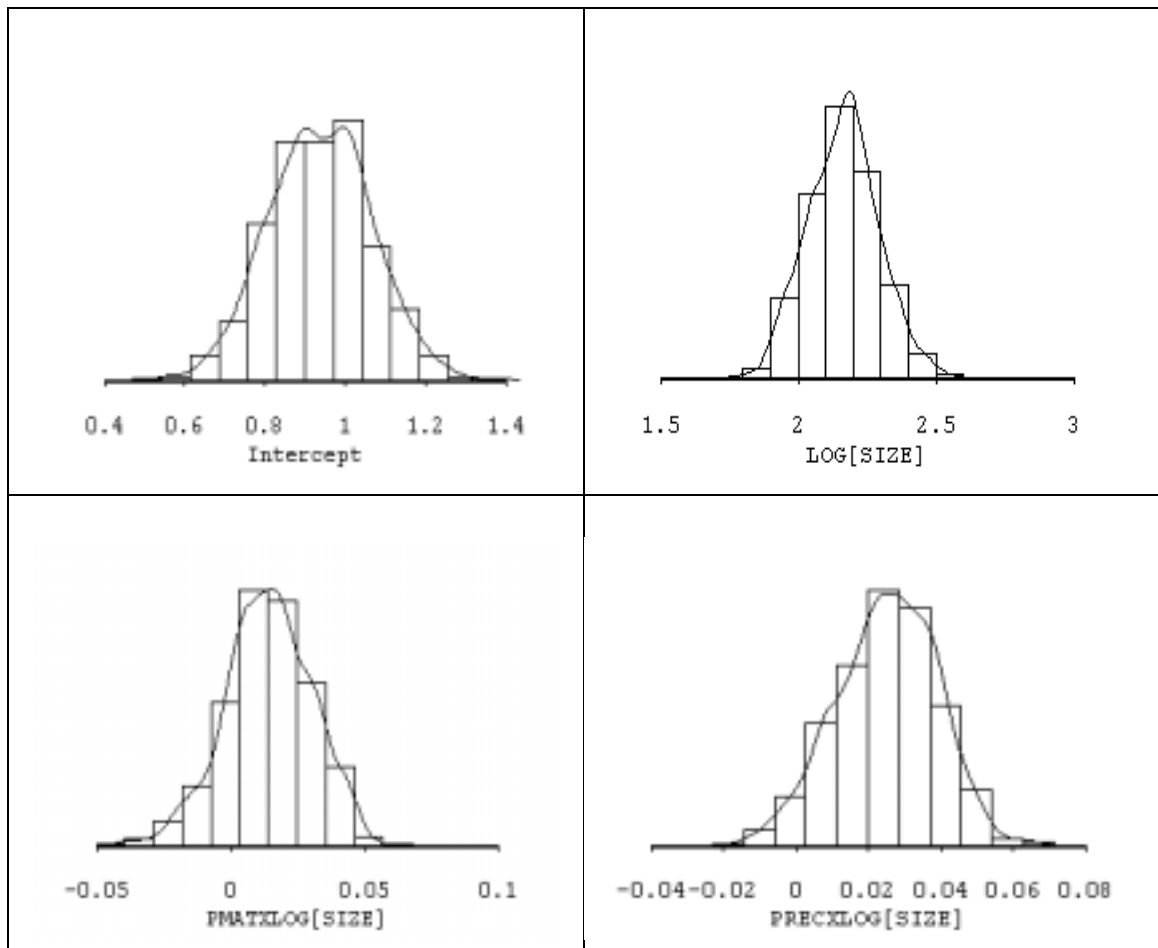


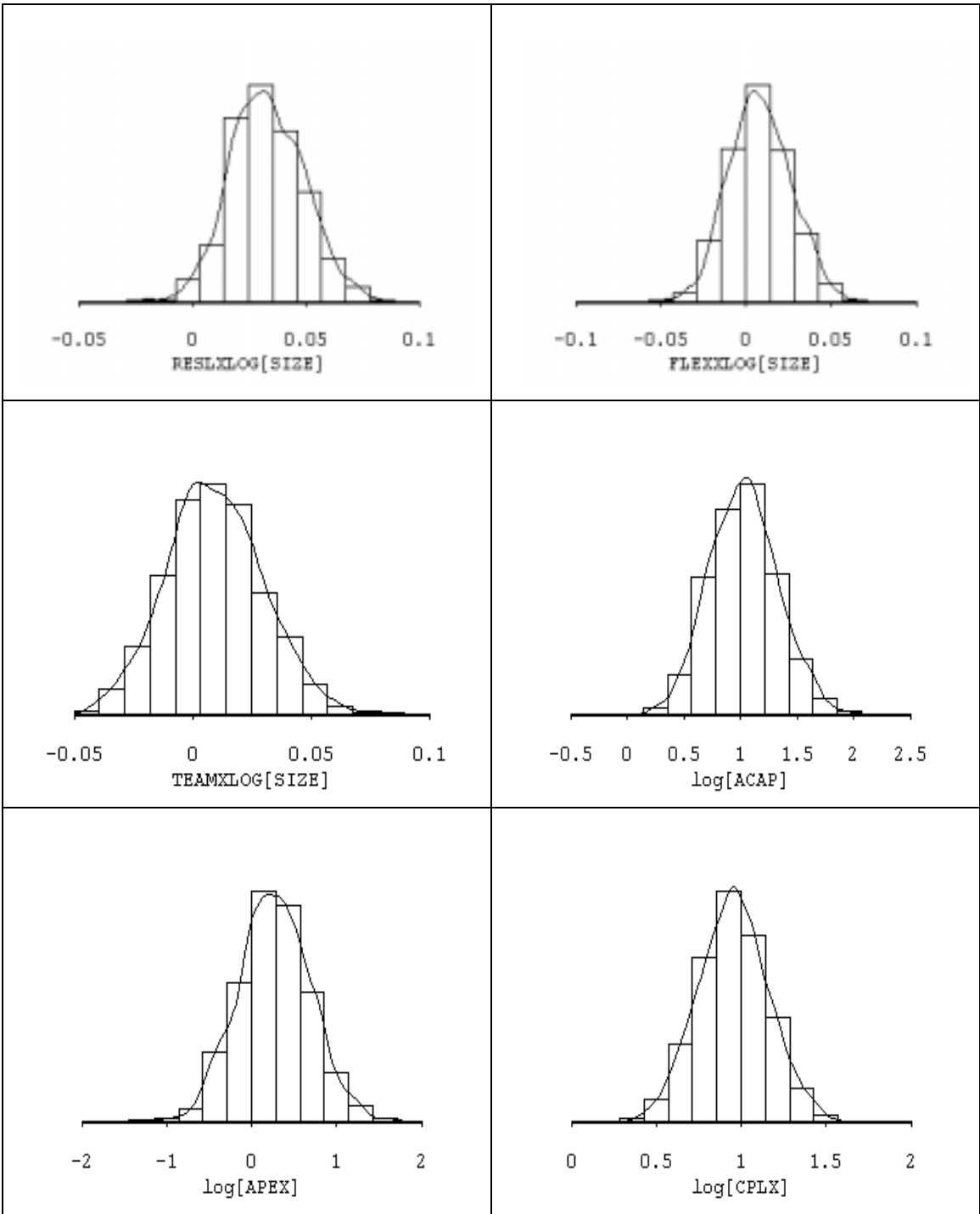


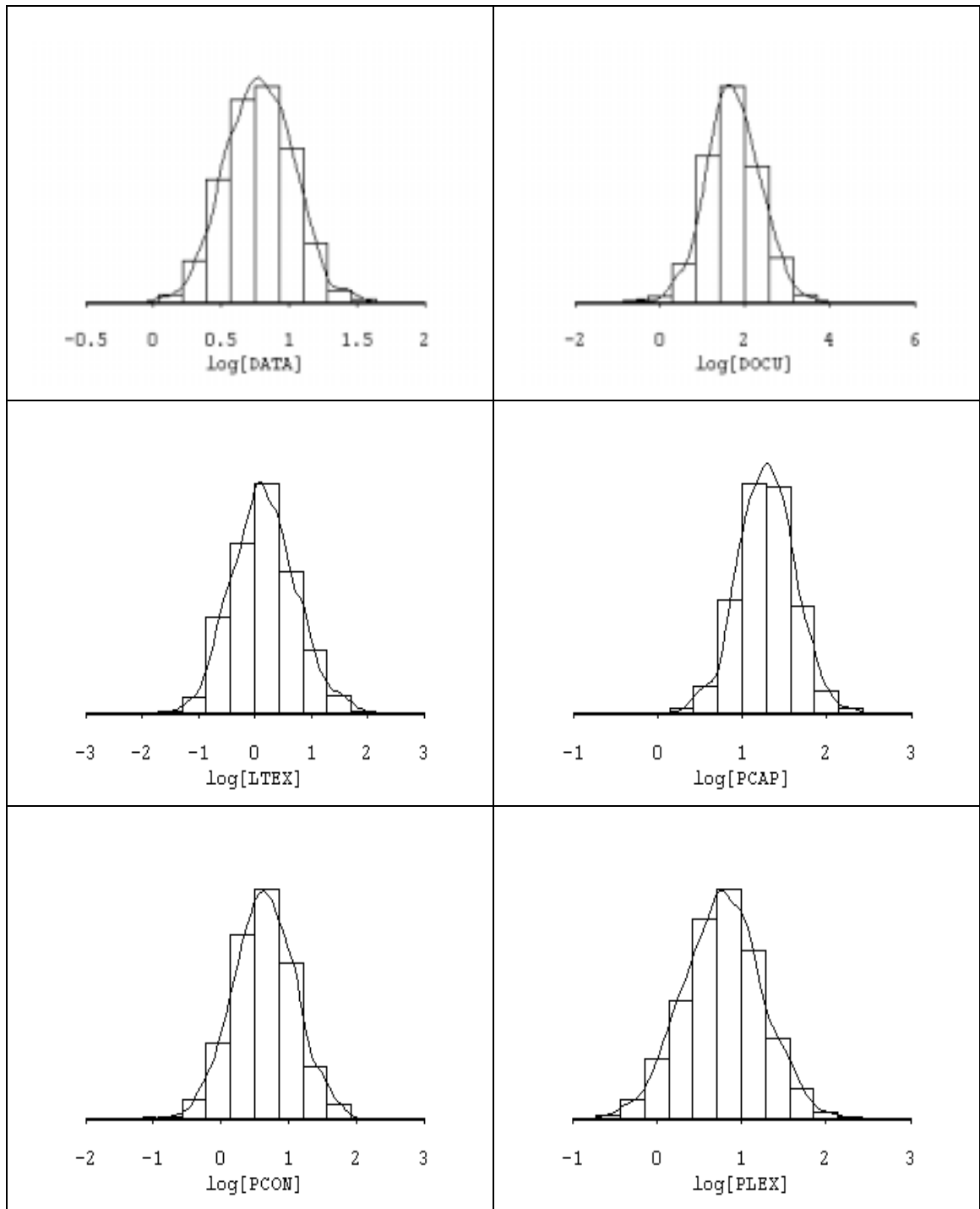


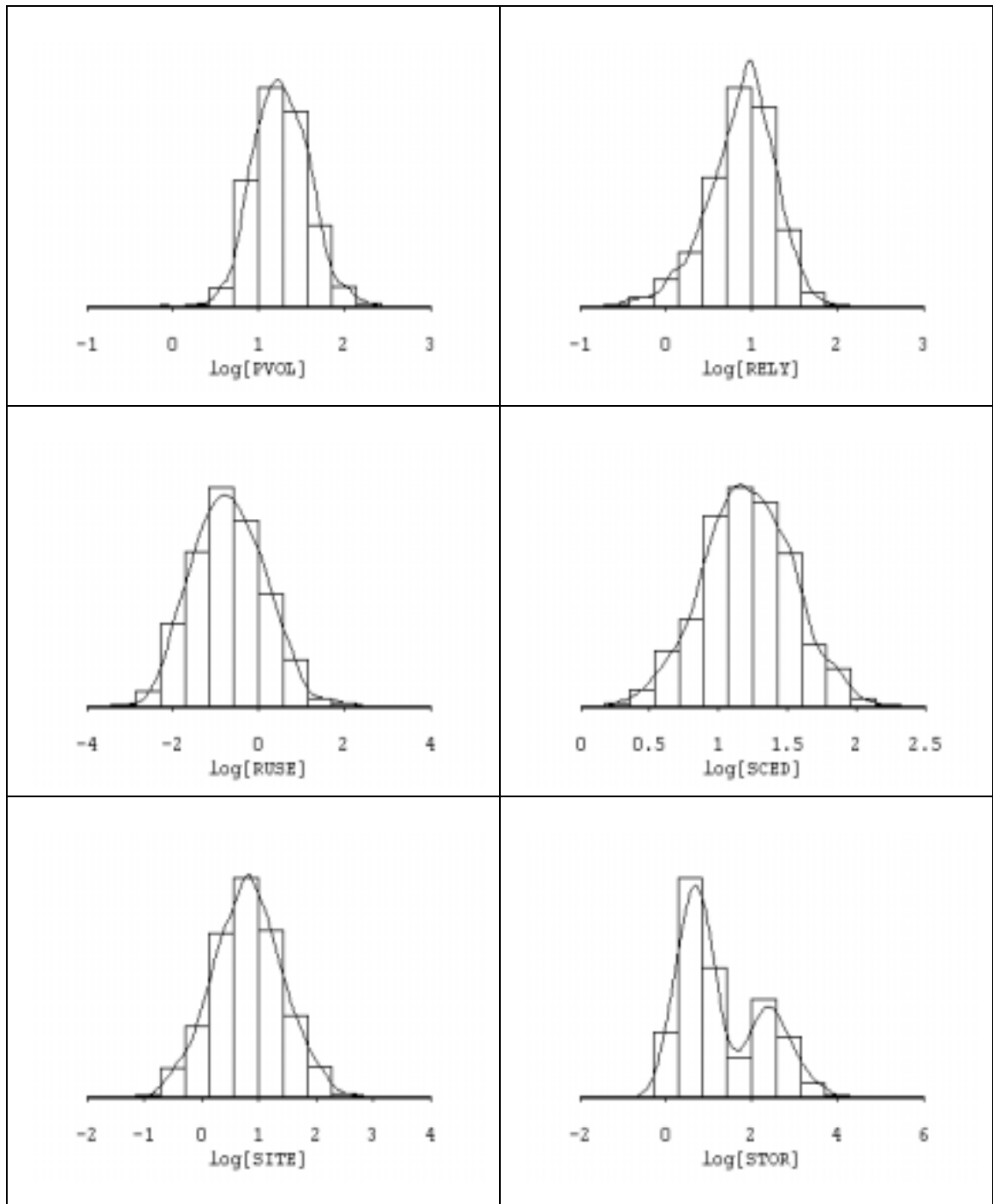


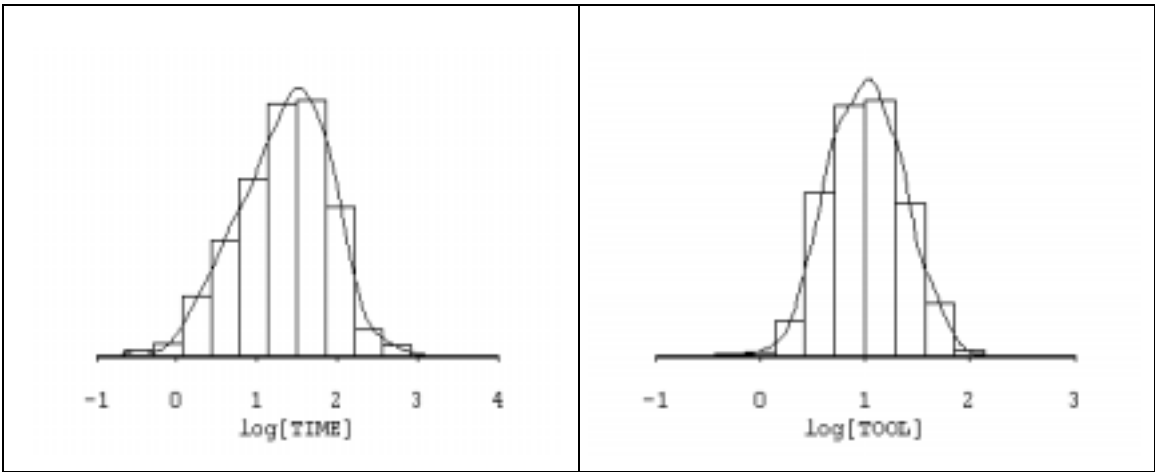
B.3 Histograms (Three dimensional TOOL)











B.4 Probability Plots (Three dimensional TOOL)

