

# CASE Impact Analysis on Software Development Effort via Bayesian Modeling Approach

Jongmoon Baik

Computer Science Department  
University of Southern California  
Los Angeles, CA 90089-0781  
[jobaik@sunset.usc.edu](mailto:jobaik@sunset.usc.edu)  
1-213-740-6470

Barry Boehm

Computer Science Department  
University of Southern California  
Los Angeles, CA 90089-0781  
[boehm@sunset.usc.edu](mailto:boehm@sunset.usc.edu)  
1-213-740-8163

**Abstract.** Over the last two decades, CASE (Computer Aided Software Engineering) tools have led to the improvement of software productivity and quality by replacing human tasks with automated support in software development processes. Many initiatives in the field were pursued in the 1980's and the early 1990's in order to provide more effective CASE technologies and environments. Most software estimation models developed during the last two decades focuses only on functionalities supported by CASE tools to estimate their impacts on software development effort without considering other important factors such as tool integration, tool maturity, and user support. In this paper, we provide a extended set of tool rating scales from COCOMO (COConstructive COSt MOdel) II which is one of widely used software estimation model. We report an analysis result of CASE tool effects on software development effort obtained from Bayesian approach.

## 1. Introduction

The proliferation of CASE tools have enabled software development teams to use a variety of CASE tools for their projects with the hope of delivering—on-time by improving software productivity. However, They use software tools that are assembled over time and adopt new tools without establishing a set of formal evaluation criterion. Several researches [1][2] provide guidelines for the assessment of CASE tools and several parametric software cost estimation model make “use of software tools” as one of the environmental factors that affects the software productivity in order to assess the impact of CASE tools. While most of those software estimation models are proprietary, COCOMO (Constructive Cost Model) [3] is one of the available and widely accepted models in public and has been updated to COCOMO II to address the issues such as non-sequential and rapid development models; reuse-driven approach including COTS (Commercial-Off-The-Shelf) packages, reengineering, application composition, and application generation capabilities; object-oriented approaches supported by distributed middleware; and software maturity effects [4]. The latest version of COCOMO II gives fairly good prediction rates that are within 20% of actuals 56% of the time, within 25% of actuals 65% of the time, and within 30% of the actuals 71% of the time [5].

COCOMO II uses TOOL, one of the Effort Multipliers in the model, to capture the influence on software productivity and provides a TOOL rating scale as a guideline for the evaluation of software tools. However, even if COCOMO II TOOL rating scale was updated with 1990's software tools from the original COCOMO TOOL rating scale, it may not be complete without considering other important factors such as tool integration, tool maturity and user support. Many software tool evaluation criteria [1][6][7] consider the degree of tool integration in the production process, tool maturity in the market, and services provided by vendors as important factors in assessing the software tools in addition to the quality of tools. For these reasons, this paper provides a extended set of COCOMO II TOOL rating scale that reflect those important factors with currently available tool sets. This paper also shows the analysis result of a mathematical model developed to effectively identify the impact of the extended three dimensional COCOMO II TOOL rating scales on projects' effort estimation by using Bayesian approach. For the validation of the extended research model, two of cross validation methodology (Cross Validation by Data-Splitting & Bootstrapping).

## 2. COCOMO II and Tool Productivity Impact

The COCOMO II is one of the most popular software cost estimation models. It has been updated from the original COCOMO [3] and its Ada successor [8] in order to address issues on new process models and capabilities such as concurrent, iterative, and cyclic processes; rapid application development, COTS integration and software maturity initiatives. It provides a tailorable mix of three sub-models such as Application Composition model, Early Design mode, and Post Architecture model, which are consistent with the granularity of the available information to support software estimation. Among the three models, the Post Architecture model is the most mature and detailed model that can be used when the system architecture is developed. It predicts the effort estimation, Person Month (PM) and the schedule in calendar months as the following:

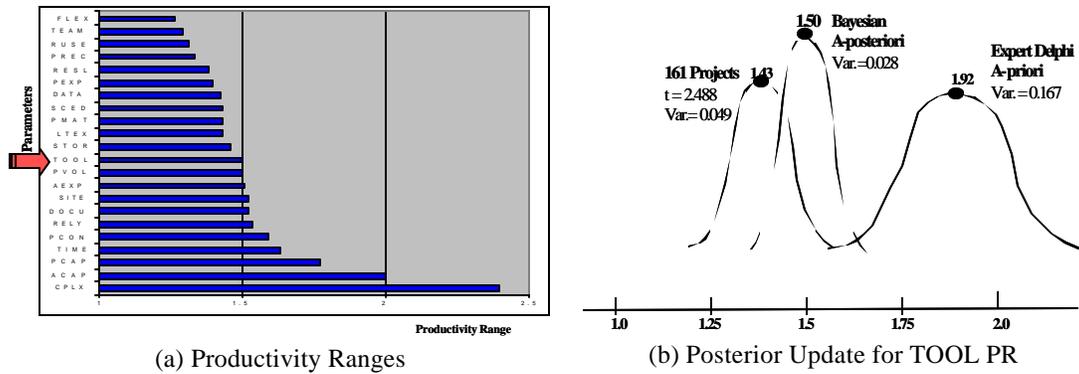
$$PM = A \cdot (Size)^{B+0.01 \sum_{i=1}^5 SF} \cdot \prod_{j=1}^{17} EM_j \quad \text{EQ. 1}$$

The key inputs of the model are Software size being developed; 5 scaling factors, 17 Effort Multipliers. Scaling Factors and Effort multipliers are listed in Table 1:

Scaling Factors	Effort Multipliers	
Precedentedness (PREC) Development Flexibility (FLEX) Architecture/Risk Resolution (RESL) Team Cohesion (TEAM) Process Maturity (PMAT)	<b>Product:</b> Required reliability (RELY) Database size (DATA) Documentation match to life-cycle needs (DOCU) Product complexity (CPLX) Required reusability (RUSE)	<b>Platform:</b> Time constraint (TIME), Storage constraint (STOR) Platform volatility (PVOL)
	<b>Personnel:</b> Analyst capability (ACAP) Applications experience (APEX) Programmer capability (PCAP) Platform experience (PLEX) Language and tool experience (LTEX) Personnel continuity (PCON)	<b>Project:</b> Use of software tools (TOOL) Required development schedule (SCED) Multi-site development (SITE)

**Table 1. COCOMO II Scale Factors and Effort Multipliers**

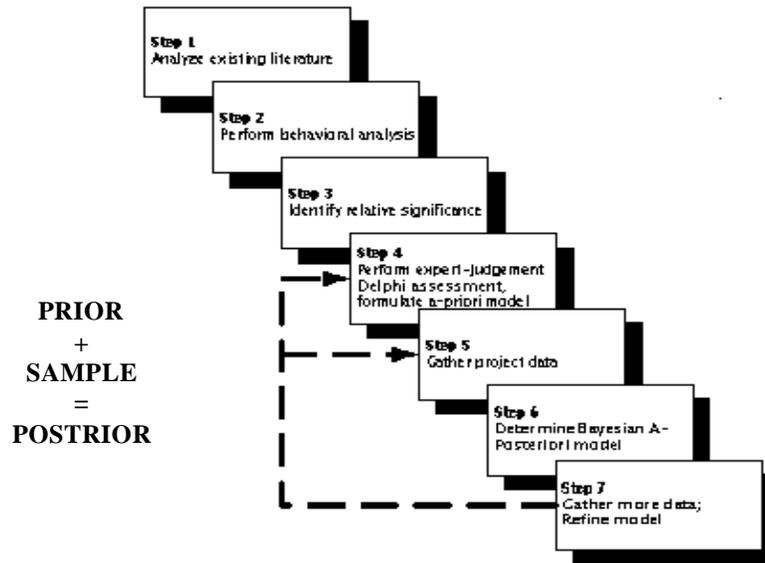
COCOMO II has been updated from the original COCOMO and Ada COCOMO to address the issues on new process models and capabilities such as concurrent, iterative, and cyclic processes; rapid application development, COTS (Commercial-Off-The-Shelf) integration and software maturity initiatives. It provides a tailorable mix of three sub models: 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. Among those sub models, The Post Architecture model is the most mature and detailed model that estimates the effort and the schedule involved in the development and maintenance of a software product. The COCOMO II Post Architecture model uses TOOL, one of Effort Multipliers, which captures productivity impact on software development effort. Figure 1.a shows that the Productivity Range (PR: ratio between the least productivity parameter rating and the most productive parameter rating) for TOOL (1.5) is a relatively high payoff in a software improvement activity among 22 COCOMO II Post Architecture parameters. Figure 1.b shows how the PR of TOOL rating scale was determined in the COCOMO II.2000 Bayesian approach. Also, the PR of TOOL has statistically a strong significance (t-value = 2.488 > 1.96) in estimating the software development effort. That is, increasing tool rating makes a statistically significant difference in reducing software development effort, even after normalizing the effects of other variables such as process maturity (PMAT). Given that previous anecdotal data still generates controversy on this point, this is valuable result.



(a) Productivity Ranges (b) Posterior Update for TOOL PR  
**Figure 1. Bayesian Productivity Range**

### 3. COCOMO II Seven-Step Modeling Methodology

The COCOMO II and other related models like COQUALMO Constructive QUALity Model) [9] uses the seven-step modeling methodology shown in Figure 2 that support the development of parametric research models. This methodology is adopted to develop the research model that explains CASE tool effect on the production process effort.



**Figure 2. COCOMO II 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 variable) 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 factors on project activity levels.

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

After a thorough behavioral analyses was done, the relative significance and orthogonality of each tool factor was qualitatively confirmed, and rating scales were determined for each factor. Also, a candidate functional form for a more robust COCOMO II TOOL parameter was determined

$$\begin{aligned}
\text{TOOL} &= b_1\text{TCOV} + b_2\text{TINT} + b_3\text{TMAT} \\
b_1, b_2, b_3 &\geq 0 \\
b_1 + b_2 + b_3 &= 1
\end{aligned}
\tag{EQ. 2}$$

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

Here, cost estimation experts from among USC’s COCOMO II Affiliates were convened in a two-round Delphi exercise to determine a-priori values of the weights  $b_1$ ,  $b_2$ , and  $b_3$ . This model version is based on expert-judgement and is not calibrated against actual project data. But it serves as a 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 6 shows the resulting regression analysis.

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

Using the expert-determined drivers and their variances as a-priori values, we determined a Bayesian a-posteriori set of model parameters 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, including covariance effects.

**Step 7:** Gather more data to refine the model.

Continue to gather data, and refine the model to be increasingly data-determined vs. expert-determined. We are gathering TCOV, TINT, and TMAT ratings on future projects entered into the COCOMO II database.

#### 4. Extended CASE Tool Rating Scales by Productivity Dimensions

As mentioned earlier, most software estimation models assess CASE tools base on only their functionalities. Here we introduce the extended CASE tool rating scales from COCOMO II one-dimensional TOOL rating scale.

##### *Completeness of Tool 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 [10]. However, it is necessary to partition the same kind of tools by their functionalities to explain differences in the productivity impact. This section provides an extension of the COCOMO II TOOL rating scale with currently available CASE tools in the software market.

Table 2 provides a classification of currently available tool sets based on how completely they support the given activities in software development processes and how effectively they support CMM (Capability Maturity Model) Tool characteristics at each level developed by SEI (Software Engineering Institute) [11][12]. Like COCOMO II TOOL rating scale, it also requires a certain amount of judgment 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
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 2. Rating Scale for the Completeness of Tool Coverage**

### *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 [13]. 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 a new system, and software reuse can be obtained [6]. Thereby, effort and schedule for the software development can be decreased.

Table 3 shows another dimension of the CASE tool rating scale to evaluate how well tools are integrated and what kinds of mechanisms are provided to exchange information among tools in an integrated environment. This classification is based on Wasserman's five level models for tool integration in software development environments [14]. 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.

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

<b>Rating</b>	<b>TINT</b>
Very Low	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
Low	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
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
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

**Table 3. Rating Scale for the Degree of Tool Integration**

**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 the tool set is 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 set and consequently the more effort is required to correct those errors. Table 4 categorizes CASE tools according to the survival length in the software market after they are released.

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

**Table 4. Rating Scale for Tool Maturity and User Support**

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 [2][7]. These evaluation criteria rate tools quantitatively according to tool supports provided by software vendors. Likewise, CASE tools are categorized in a scale, ranging from Very Low to Very High according to tool’s user support.

**5. Bayesian Analysis**

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. A posterior probability is defined according to the precision of a-priori (Delphi or Expert-judged) and sample (Data-determined) information by using Bayes’ theorem[15]. In order to get a consensus-based relative weighting value for each of the extended rating scales, two round Delphi process were carried out. For this Delphi process, participants were selected from USC-CSE Affiliates. For sample information, 15 project data points that have information about the three rating scales were adopted from the COCOMO II 161 project data used for COCOMO II.2000 calibration[4]. In order to determine the weighting values for the extended set of TOOL rating scales, the regression model shown in EQ. 2,  $TOOL = b_1TCOV + b_2TINT + b_3TMAT$  ( $b_i$  non-negative and summing to 1), was used to obtain the weight values. The sample weighting values were obtained by regression analysis

$$Var(b^{**}) = \frac{1}{s^2} \frac{1}{X^T X + H^*} \frac{1}{s^2} \frac{1}{X^T X + H^*} X^T Y$$

supported by the program Arc[16]. In order to get posterior mean ( $b^{**}$ ) and variance ( $Var(b^{**})$ ) for the weighting values, the following equations are used[17].

**EQ. 3**

Where X is a matrix of observations on predictor variables, s is the variance of the residual for the sample data, and  $b^*$  and  $H^*$  are the mean of prior information and inverse of prior variance matrix, respectively. Table 5 shows the mean and variance distributions for prior, sample, and posterior weighting values for the extended three-dimensional tool-rating scales. As shown in the table, the sum of posterior weighting value mean is not 1. Therefore, TOOL rating value in the research model can be determined by using normalized posterior coefficient means as  $TOOL = 0.51 * TCOV + 0.27 * TINT + 0.22 * TCOV$ . 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.

Distribution		$b_1$	$b_2$	$b_3$
PRIOR	MEAN	0.47	0.26	0.27

	VARIANCE	0.025694	0.005485	0.016875
SAMPLE	MEAN	0.515982	0.282561	0.165480
	VARIANCE	0.0078697	0.011590	0.012409
POSTRIOR	MEAN	0.495104	0.259691	0.2111617
	VARIANCE	0.00461028	0.00335019	0.005255464

**Table 5. Prior, Sample, and Posterior Distributions**

In order to compare the model with the COCOMO II Bayesian estimation result, an evaluation criterion, the percentage predictions that fall within X% of actuals denoted as PRED(X), is used. The models are evaluated at PRED(0.10), which is done by counting the number of MRE (Magnitude of Relative Errors)[18] in the below less than or equal to 0.10 and dividing the number of projects. Table 2 summarizes the prediction accuracies of COCOMO II, Sample without prior information, and Posterior with prior and sample. Both Sample and Posterior with the three dimensional rating scales gives better prediction accuracies than in COCOMO II.

	COCOMO II	Sample	Posterior
Rating Scale	TOOL	TCOV, TINT, & TMAT	
PRED(.10)	67%	87%	87%

**Table 6. Comparison of Prediction Accuracies**

## 6. Cross –Validation

### Data-Splitting

Cross validation by data-splitting divides the original dataset into two sub samples: *Construction set* for exploration and model formulation, *Validation set* for model validation, formal estimation, and testing. In order to validate the research model by data-splitting[19], we used the Arc statistical analysis program on two datasets of the same size (161 project data). For the TOOL rating value of the first dataset, the standard COCOMO II one-dimensional TOOL rating scale was used. In the second dataset, the TOOL rating value of the 15 projects with TCOV, TINT, and TMAT ratings was determine by the Bayesian weighted sum of the TCOV, TINT, and TMAT ratings. The TOOL ratings of the rest of the project data have the same values as in the first dataset. For the two cross validation, we used the 115 project data points for the construction sets for the regression model formation. The remaining, randomly chosen 46 project data points were assigned into the validation dataset for regression model validation. Of those 15 projects, 11 were in the construction dataset and 4 were in the validation set. Table 7 shows the cross validation results.

1 <sup>st</sup> validation set (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
2 <sup>nd</sup> Validation set (TCOV, TINT, & TMAT)
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 7. Cross Validation Summary and PRESS**

In the above table, 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[20]. 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, predicted residual sum of squares, (0.260493) from the validation dataset with the extended TOOL rating scales is also smaller than the PRESS (0.262203) from the validation dataset with the one dimensional TOOL rating scale. 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.

### Bootstrapping

The bootstrap is a statistical simulation methodology that re-samples from the original data set[21]. This methodology has been used 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 in wider application areas such as nonlinear regression, logistics regression, spatial modeling, and so on. We used the bootstrap option in the *Arc* program to run 1000-case bootstrap samples for both the one-dimensional and three-dimensional TOOL ratings, by randomly sampling with replacement from 161 project data points. The detailed information about the bootstrap supported by *Arc* is available in [22]. Table 8 shows the standard error and bias from the two bootstrap objects. The standard error estimates compare with the usual estimates obtained from the normal regression with the two datasets, respectively. The bias is difference between the average of bootstrap estimates and normal estimate for each variable. 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
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[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 8. Bootstrap Std-Error and Bias**

Table 9 shows the corresponding percentile bootstrap confidence intervals for the coefficient estimates of Effort Multipliers. 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 from the two datasets, respectively, the percentile bootstrap confidence intervals indicates that the re-sampling results agree closely with those obtained from standard methods.

Coefficients	One-dimensional TOOL	Three-dimensional TOOL
log[ACAP]	(0.434194 1.67577)	(0.449679 1.64775)
log[AEXP]	(-0.689046 1.11136)	(-0.505828 1.0968)
.	.	.
.	.	.
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 9. Confidence Intervals of the Parameters**

Once again, most of the percentile confidence intervals from the three dimensional TOOL data set are narrower than those obtained from the one-dimensional dataset. Especially, the confidence interval for the coefficient of  $\log[\text{TOOL}]$  from the three-dimensional dataset has a smaller region than that from the one-dimensional dataset. 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.

## 7. Conclusion

In this paper, a set of multi dimensional CASE tool rating scale, which is extended from the COCOMO II TOOL rating scale and reflect important CASE tool environmental factors that have impacts on software development effort such as the degree of tool integration, tool maturity, and user support. It gives a guideline to effectively evaluate CASE tools as it does in the COCOMO II one dimensional TOOL rating which focuses on only the completeness of too coverage. This paper also introduces a method to calibrate the individual contribution to a multi dimensional parameter. To find a best set of the weighting values for the three rating scales, we use the Bayesian approach to combine two source of information (expert-judged and data-determined). Thereby, the prediction accuracy is fairly improved from 67% to 87% actuals with PRED (.10) over 15 project data. Two cross validation results adopted from *Arc* show that the regression fit with the three dimensional tool rating scales is somewhat better than with the one dimensional TOOL rating scale, and is stable with respect to the other COCOMO II parameters.

The particular values should be considered provisional, as they are based on only 15 data points with detailed TOOL component ratings. But, since the t-values from the regression are strong, and since the relative magnitudes of the values exhibit consistency between the expert-determined and data-determined values, we find sufficient justification to add the TOOL component parameters to the COCOMO II model. We plan to refine the parameter values as more data is collected

## References

- [1] Robert Firth et. al, A Guide to the Classification and Assessment of Software Engineering Tools, CMU/SEI-TR-10, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1987.
- [2] Vicky Mosley, "How to Assess Tools Efficiently and Quantitatively", IEEE Software, May 1992.
- [3] Barry W. Boehm, Software Engineering Economics, Prentice Hall, 1981.
- [4] Barry W. Boehm et. al., Software Cost Estimation with COCOMO II ,Prentice Hall, July 2000.
- [5] Sunita Chulani et. al., " Bayesian Analysis of Empirical Software Engineering Cost Models", IEEE Transaction on Software Engineering, pp. 513-583, July-Aug 1999.
- [6] Ian Sommerville, Software Engineering, 5<sup>th</sup> Edition, Addison-Wesley Pub Co., 1995.
- [7] Contents and Services, Ovum Ltd., UK, Jun 1995.
- [8] Barry W. Boehm and Walker Royce, "Ada COCOMO and Ada Process Model", Proceedings of 5<sup>th</sup> COCOMO User's Group Meeting, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1989.
- [9] Sunita Chulani and Barry W. Boehm, Modeling Software Defect Introduction Removal: COQUALMO (Constructive QUALity Model), USC-CSE-99-510, The Center for Software Engineering, University of Southern California, Los Angeles, CA, 1999.
- [10] Report on Project Management and Software Cost Estimation Technologies, STSC-TR-012, System Technology Support Center, Apr 1995.
- [11] Paulk, M. C. et. al., Capability Maturity Model for Software, Version 1.1, CMU/SEI\_TR\_25, Carnegie Mellon University, Pittsburgh, PA, 1993.

- [12] Alan M. Christie, A Practical Guide to the Technology and Adoption of Software Process Automation, CMU/SEI-94-007, Carnegie Mellon University, Pittsburgh, PA, 1994.
- [13] David Sharon and Rodney Bell, TOOLS THAT BIND: Creating Integrated Environments, IEEE Software, pp. 76-85, Mar 1995.
- [14] Wasserman, A. I., Tool Integration in Software Engineering Environments, In Proc. Int. Workshop on Environments, Berlin, pp137-149, 1990.
- [15] George G. Judge et. al., The Theory and Practice of Econometrics, 2<sup>nd</sup> Edition, John Wiley and Sons, 1985.
- [16] Dennis Cook and Sanford Weisberg, Applied Regression Including Computing and Graphics, Wiley Series, 1999.
- [17] Edward E. Leamer, Specification Searches: Ad hoc Inference with Nonexperimental Data, Wiley Series, 1978.
- [18] Conte, S. et. al, Software engineering Metrics and Models, Benjamin/Cummings, Menlo Park, CA, 1986.
- [19] John Neter et. al., Applied Linear Regression models, 3<sup>rd</sup> Edition, IRWIN, 1996.
- [20] Snaford Weisberg, Cross-Validation in Arc,  
<http://www.stat.umn.edu/arc/crossvalidation/crossvalidation/crossvalidation.html>, Dec. 1999.
- [21] Michael R. Chernick, Bootstrap methods: A Practitioner's Guide, John Wiley & Sons, Inc., 1999.
- [22] Iain Pardoe, "An Introduction to Bootstrap Methods using Arc". Technical Report 631, University of Minnesota, St. Paul, MN 55108, Feb 2000.