한국산업정보학회논문지 제5권 제2호 (2000.6)

An Improved Calibration Method for the COCOMO II Post-Architecture Model

Myoung-Young Yoon*

요 약 개발하는 소프트웨어의 비용, 스케쥴 및 노력을 예측하기 위하여 오늘날 많은 소프트웨어 비용 모델이 개발되었다. COCOMO II은 비순차적이며, 빠른 개발방법 과정 등의 새로운 소프트웨어 생명주기에 적합한 비용 모델이다. COCOMO II 모델에서 최소자승 회귀분석 방법은 조율 방법으로 널리 사용되었다. 전통적인 회귀분석 조율 방법은 데이터 셀에 대한 가정이 위배된다. 즉, 원시자료는 특히 서로 다른 개발조직으로부터 비용인자 등급, 노력, 크기가 수집되며 부정확하고 이상치가 존재한다. 본 논문에서는 이러한 한계를 극복하기 위하여 우리는 COCOMO II 모델을 가지고 상대오차를 최소화하는 모델 조율에 대한 새로운 방법을 제안한다. 제안된 방법의 특징은 이상치를 갖는 원시 데이터에 덜 민감한 특성을 갖고 있다. 실험결과, 제안된 새로운 조율방법 MRE가 조정된 결정계수($adj-R^2$), 표준편차($\hat{\sigma}$), 예측 정도($PRED(L)$))에서 기존의 전통적 회귀분석 방법보다 우수하게 나타났다.

Abstract To date many software engineering cost models have been developed to predict cost, schedule, and effort of the software under development. The COCOMO II is well- suited for the new software development life cycle such as *non-sequential and rapid- development processes*. The traditional regression approach based on the least square criterion is the most commonly used technique for empirical calibration in the COCOMO II model. It has a few assumptions frequently violated by software engineering data sets. The source data is also generally imprecise in reporting size, effort, and cost-driver ratings, *particularly across different organizations. And that the outlier for the source data is a* peculiarity and indicates a data point. To cope with difficulties, in this paper, we propose a new regression method for calibrating COCOMO II post-architecture model based on the minimum relative error(MRE) criterion. The characteristic of the proposed method is insensitive to the extreme values of the data *in the empirical calibration. As the experimental* results, It is evident that our proposed calibration method MRE was shown to be superior to the traditional regression approach for model calibration, as illustrated by the values obtained for standard deviation($\hat{\sigma}$), and prediction at level L $PRED(L)$ measures.

1. Introduction

One of the software engineering's longstanding problems is to estimate software cost. Software is the most expensive element in many computer based systems and a large cost estimation error can make the difference between profit and loss. Software development costs continue to increase and practitioners continually express their concerns over there inability to accurately predict the costs involved. Thus, one of the most important objectives of the software engineering community has been to develop useful models that continually explain the software development life-cycle and accurately predict the cost of developing a software product. To that end, many parametric software estimation models have evolved in the last two decades[1, 2, 3, 4]. The COCOMO and Ada COCOMO cost models have experienced difficulties in estimating software projects of the 90's due to new practices such as non-sequential and rapid-development process models; reuse-driven approaches involving commercial-off-the-shelf packages, reengineering, application composition, and application generation capabilities; object-oriented approaches supported by distributed middleware; software process maturity effects and process-driven quality estimation. The rapidly changing nature of software development has made it extremely difficult to develop empirical models that continue to yield high prediction accuracies.

* 충청대학 컴퓨터학부 부교수

The COCOMO II model is new functional forms reflecting these practices and is well-suited for the new software life cycle processes and capabilities. And that it incorporated proven features of COCOMO 81 and Ada COCOMO models. The COCOMO II model would be useful for the next generation of software development [3]. We will calibrate the COCOMO II Post-Architecture model in order to accurately predict the cost of the development software. The new model COCOMO II which consists of three models is recent update of the popular COCOMO model published in[1]. The Application Composition model is used to estimate effort and schedule on projects. The Early Design model is used when a rough estimate is needed based on incomplete project and product analysis. The Post-Architecture model is used when top level design is complete and detailed information is known about the project.

Almost all of the above mentioned parametric models have been empirically calibrated to actual data from completed software projects. The most commonly used technique for empirical calibration has been the popular traditional regression approach. The traditional method employed by elementary statistic books and most canned statistical software is that of least squares estimation. In this case the fitted line is chosen so that the sum of the squared differences between an observed value, y_i , of the dependent variable and the predicted value $\hat{y}_i$, is minimized, i.e., min $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$. This is called least squares(LS) estimation. The traditional regression approach a few assumptions frequently violated by software engineering data sets. The source data is also generally imprecise in reporting size, effort, and cost-driver ratings, particularly across different organizations. And that the outlier for the data source is a peculiarity and indicates a data point which is not at all typical of the rest of the data. This results in the development of inaccurate empirical models that do not perform very well when used for prediction.

To cope with difficulties, in this paper, we review the calibration of the Post-Architecture model and propose a new method for calibrating COCOMO II Post-Architecture model based on the minimum relative error(MRE) criterion. Section 2 of this paper review the process of the COCOMO II Post-Architecture model to be calibrated. In section 3, we propose the MRE regression technique for calibrating Post-Architecture model. Section 4 describes the data used for calibration, the calibration procedures, results, and the model forecast accuracy. Section 5 has our conclusions and remarks.

2. Post-Architecture Model

The COCOMO II Post-Architecture model is fully described in [5, 6]. The Post-Architecture model covers the actual development and maintenance of a software product. This stage of the life cycle proceeds most cost-effectively if a software life-cycle architecture has been developed; validated with respect to the system's mission, concept of operation, and risk; and established as the framework for the product.

The Post-Architecture model predicts software development effort, person Months(PM), as shown in equation(1), and schedule in months. The model has the form:

$$PM = k \cdot (Size)^{1.01 + \sum_{i=1}^{5} SF_i} \cdot \prod_{j=1}^{17} EM_j \qquad (1)$$

It has about the same granularity as the COCOMO 81 and Ada COCOMO models. It uses source instructions and/or function points for sizing, with modifiers for reuse and software breakage; a set of 17 multiplicative cost drivers (EM); and a set of 5 scaling cost drivers to determine the project's scaling exponent (SF).

All of the cost drivers are described in <Table 1>. These scaling cost drivers replace the development modes(Organic, Semidetached, or Embedded) in the original COCOMO 81 model, and refine the four exponent-scaling factors in Ada COCOMO. Effort multipliers are grouped into four categories: product, platform, personnel, and project. Each driver can accept one of six possible ratings: Very Low(VL), Low(L), Nominal(N), High(H), Very High(VH), and Extra High(XH).

<Table 1>. COCOMO II Cost Drivers

Abr.	Sym.	Name and Description
PREC	SF_1	Precedence
FLEX	SF_2	Development Flexibility
RESL	SF_3	Architecture and Risk Resolution
TEAM	SF_4	Team cohesion
PMAT	SF_5	Process Maturity
RELY	EM_1	Required Software Reliability
DATA	EM_2	Data Base Size
CPLX	EM_3	Product Complexity
RUSE	EM_4	Required Reusability
DOCU	EM_5	Documentation
TIME	EM_6	Time Constraint
STOR	EM_7	Storage Constraint
PVOL	EM_8	Platform Volatility
ACAP	EM_9	Analyst Capability
PCAP	EM_{10}	Programmer Capability
AEXP	EM_{11}	Applications Experience
PEXP	EM_{12}	Platform Experience
LTEX	EM_{13}	Language and Tool Experience
PCON	EM_{14}	Personnel Continuity
TOOL	EM_{15}	Use of Software Tools
SITE	EM_{16}	Multi-Site Development
SCED	EM_{17}	Required Development Schedule

The selection of scale factors (SF) in equation (1) is based on the rationale that they are a significant source of exponential variation on a project's effort or productivity variation. Software cost estimation models often have an exponential factor to account for the relative economies or diseconomies of scale encountered in different size software projects. The scale factor constant, 1.01, is set to a value greater than one because it is believed that scale factors exhibit diseconomies of scale. This is generally due to two main factors: growth of interpersonal communications overhead and growth of large-system integration overhead [7]. Recent research has confirmed diseconomies of scale influence on effort with the Process Maturity (PMAT) scaling cost driver [8]. An econometrics log-log model and the COCOMO II Post-Architecture model were used to determine effect on effort.

<Table 2> shows the *a priori* values assigned to each rating before calibration. All cost drivers are qualitative, except for size, and are measured by selecting one of six ratings: Very Low (VL), Low (L). Nominal (N), High (H), Very High (VH). and Extra High (XH). As can be seen in <Table 2> not all six rating levels were valid for all cost drivers. An example of this is the lack of a VL rating for the RUSE cost driver.

<Table 2>. A priori Model Values

Driver	Sym	VL	L	N	H	VH	XH
PREC	SF_1	0.05	0.04	0.03	0.02	0.01	0.0
FLEX	SF_2	0.05	0.04	0.03	0.02	0.01	0.0
RESL	SF_3	0.05	0.04	0.03	0.02	0.01	0.0
TEAM	SF_4	0.05	0.04	0.03	0.02	0.01	0.0
PMAT	SF_5	0.05	0.04	0.03	0.02	0.01	0.0
RELY	EM_1	0.75	0.88	1.00	1.15	1.40	
DATA	EM_2		0.94	1.00	1.08	1.16	
CPLX	EM_3	0.75	0.88	1.00	1.15	1.30	1.65
RUSE	EM_4		0.89	1.00	1.16	1.34	1.56
DOCU	EM_5	0.85	0.93	1.00	1.08	1.17	
TIME	EM_6			1.00	1.11	1.30	1.66
STOR	EM_7			1.00	1.06	1.21	1.56
PVOL	EM_8		0.87	1.00	1.15	1.30	
ACAP	EM_9	1.5	1.22	1.00	0.83	0.67	
PCAP	EM_{10}	1.37	1.16	1.00	0.87	0.74	
PCON	EM_{11}	1.26	1.11	1.00	0.91	0.83	
AEXP	EM_{12}	1.23	1.10	1.00	0.88	0.80	
PEXP	EM_{13}	1.26	1.12	1.00	0.88	0.80	
LTEX	EM_{14}	1.24	1.11	1.00	0.9	0.82	
TOOL	EM_{15}	1.20	1.10	1.00	0.88	0.75	
SITE	EM_{16}	1.24	1.10	1.00	0.92	0.85	0.79
SCED	EM_{17}	1.23	1.08	1.00	1.04	1.10	

The size input to the Post-Architecture model, equation (1), includes adjustments for breakage effects, adaptation, and reuse. Size can be expressed as Unadjusted Function Points (UFP) [9] or thousands of source lines of code (KSLOC). The COCOMO II size reuse model is nonlinear and is based on research done by Selby[10]. The COCOMO II sizing model captures a non-linear effect with six parameters: percentage of design modified; the percentage of code modified; the percentage of modification to the original integration effort required for integrating the reused software; software understanding for structure. clarity, and self- descriptiveness; unfamiliarity with the software for programmer knowledge

of the reused code; and assessment and assimilation for fit of the reused module to the application[6].

3 Regression for Calibrating the Model

In this section we review the traditional regression techniques and propose the MRE regression with minimum relative error criterion so as to calibrate COCOMO II Post-Architecture model. In this regression model, the effort *PM* will appear as dependent variables and the independent variable will be cost drivers. Linear regression models are formed by choosing the best subset from a set of independent variables in order to explain as much variation in the dependent variable as possible. The traditional regression approach a few assumptions frequently violated by software engineering data sets. The source data is also generally imprecise in reporting size, effort, and cost-driver ratings, particularly across different organizations. Especially, the outlier for the data source is a peculiarity and indicates a data point which is not at all typical of the rest of the data.

To cope with difficulties, in this paper, we propose the technique for calibrating COCOMO II Post-Architecture model with the MRE regression. MRE technique attempt to fit these variables to sample data. Coefficients for the independent variables are produced by the traditional method LS, RLS, and the proposed MRE regression technique. These coefficients were used to adjust the previously assigned expert-determined model values.

3.1 RLS Regression Analysis

The relative least squares(RLS) method involves minimization of the sum of squared errors relative to the oserved dependent variable values[11]. Moreover, note that the RLS procedure can be formally reduced to a regular LS regression model by taking the following steps:

$$\min\left[\sum_{i=1}^{N}\left(\frac{y_i-\hat{y}_i}{y_i}\right)^2\right]=\min\left[\sum_{i=1}^{N}\left(1-\frac{\hat{y}_i}{y_i}\right)^2\right] \tag{2}$$

where $\hat{y}_i = b_0 + b_1x_1 + b_2x_2 + \ldots + b_kx_k$ and b_i are coefficients for the independent variables. By letting $y_i^* = 1$ and $\hat{y}_i^* = \hat{y}_i / y_i$, it follows that

$$\begin{aligned}\hat{y}_i^* &= (b_0 + b_1x_1 + b_2x_2 + \ldots + b_kx_k)/y_i \\ &= b_0\left(\frac{1}{y_i}\right) + b_1\left(\frac{x_{1i}}{y_i}\right) + b_2\left(\frac{x_{2i}}{y_i}\right) + \ldots + b_k\left(\frac{x_{ki}}{y_i}\right) \\ &= b_0x_{1i}^* + b_1x_{2i}^* + b_2x_{3i}^* + \ldots b_kx_{(k+1)i}^*\end{aligned} \tag{3}$$

where $x_{1i}^* = 1/y_i$, $x_{2i}^* = x_{1i}/y_i$, $x_{3i}^* = x_{2i}/y_i, \ldots, x_{(k+1)i}^* = x_{ki}/y_i$ Following substitutions, one can calculate $\min\left[\sum_{i=1}^{N}(y_i^* - \hat{y}_i^*)^2\right]$ which follows a LS minimization process and can be easily implemented.

3.2 MRE Regression Analysis

It has been found that minimization results of the average absolute error(AAE) of a fit $y=f(x)$ given by

$$AAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4}$$

provide very little information about the fit's accuracy. Levitin [12] also adds that "in order to get a better idea, one needs to relate the size of errors to the values being approximated". Furthermore, it is often useful in estimation procedures to measure the performance of a model in terms of its relative error. The average of absolute relative errors is an appropriate "loss function" for determining the quality of the prediction equation. The measure of average relative error(ARE) is defined as:

$$ARE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{(y_i-\hat{y}_i)}{y_i}\right| \tag{5}$$

ARE has been established to be useful as a measure of model performance. In the present work, however, ARE is also used as a measure of forecast quality among the various estimation techniques. In this way, uniformity is retained between the different measures of quality of fit and forecast accuracy of the estimation methods. Once a model has been fitted, the ARE of any estimation method can be easily obtained by

computing equation (5). One can derive Minimum Relative Error (MRE) estimators of the parameters of the model by solving:

$$\min\left[\sum_{i=1}^{N} |(y_i-\hat{y}_i)/y_i|\right] = \min\left[\sum_{i=1}^{N}\left|1-\frac{\hat{y}_i}{y_i}\right|\right]$$
$$= \min\left[\sum_{i=1}^{N}\left|1-\frac{b_0+b_1x_1+b_2x_2+\ldots b_kx_k}{y_i}\right|\right]$$
$$= \min\left[\sum_{i=1}^{N}\left|1-\left(\frac{b_0}{y_i}+\frac{b_1x_{1i}}{y_i}+\frac{b_2x_{2i}}{y_i}+\ldots+\frac{b_kx_k}{y_i}\right)\right|\right]$$

letting $y_i^*=1$, $\hat{y}_i^*=b_0x_{1i}^*+b_1x_{2i}^*+\ldots+b_kx_{(k+1)i}^*$

Following substitutions, one can then solve $\min\left[\sum_{i=1}^{N}|y_i^*-\hat{y}_i^*|\right]$ which is equivalent to estimating the parameters of a model using the least absolute value criterion (LAV). The MRE regression technique can be regarded as a weighted version of the LAV method with the weights equal to reciprocals of values observed.

4. Calibration of Post-Architecture Model

4.1 Data Collection

Data collection began in September 1994. The data came from organizations that were Affiliates of the Center for Software Engineering at the University of Southern California and some other sources. There organizations represent the Commercial, Aerospace, and Federally Funded Research and Development Centers (FFRDC) sectors of software development with Aerospace being most represented in the data.

Data was recorded on a data collection form that asked between 33 and 59 questions depending on the degree of source code reuse[6]. The data collected was historical, i.e. observations were completed projects. The data was collected either by site visits, phone interviews, or by contributors sending in completed forms. As a baseline for the calibration database, some of the COCOMO 1981 projects and Ada COCOMO projects were converted to COCOMO II data inputs. The total observations used in the calibration was 83, coming from 18 different organizations. This dataset formed the basis for an initial calibration.

All data that was collected was labeled with a generic identifier. The source of the data was stored in different location to project its confidentiality. The data was stored in a locked room with access restricted to project research personnel. When the data was entered into the repository, it was checked for completeness and consistency. Incomplete or inconsistent data points were dropped from the calibration dataset.

A frequent question is what defines a line of source code. Appendix B in the Model Definition Manual[11] defines a logical line of code using a framework described in [12]. However the data collected to data has exhibited local variations in interpretation of counting rules, which is one of the reasons that local calibration produces more accurate model results. Local calibration results are discussed later in this paper.

The data collected included the actual effort and schedule spent on a project. Effort is in units of Person Months. A person month is 152 hours a month and includes development and management hours. Schedule is calendar months. Adjusted KSLOC is the thousands of lines of source code count adjusted for breakage and reuse. The following three histograms show the frequency of responses for this data. Overall, the 83 data points ranged in size from 2 to 1,300 KSLOC, in effort from 6 to 11,400 person months, and in schedule from 4 to 180 months.

Data was collected on 112 projects over a two year period. From this dataset 83 projects were selected for use in model calibration. Projects with missing data or unexplainable anomalies were dropped.

4.2 Model Calibration

The calibration method adjusts all cost driver parameters in the Post-Architecture model. The statistical method we used to calibrate the model is multiple regression analysis. In this regression model, the effort *PM* will appear as dependent variables and the independent variable will be cost drivers. Coefficients for the independent variables are produced by the traditional method LS, RLS, and the proposed MRE regression technique. These coefficients were used to adjust the previously assigned expert-determined model values in Table 2.

The COCOMO model as shown in equation (6) is a

non-linear model. This means that there are values to be derived in the exponent portion of the model. To solve this problem we transform the non-linear model in equation6 into a linear model with three steps. The first step is to expand equation (1) into unique factors. Each factor will be calibrated.

$$PM = k \cdot (Size)^{1.01} \cdot (Size)^{SF_1} \cdot \cdots \cdot (Size)^{SF_5} \cdot EM_1 \cdot EM_2 \ \ldots \ EM_{16} \cdot EM_{17} \quad (6)$$

The second step is to heuristically set the values of the exponential and multiplicative cost drivers. These values are the *a priori* values shown earlier in Table 2. The third step is to transform both sides of the multiplicative form of equation (6) into a linear form using logarithms to the base *e*. This technique is called a log-log transformation[15].

$$\ln(PM \div Size^{1.01}) = \beta_0 + \beta_1 \cdot SF_1 \cdot \ln(Size) + \ldots + \beta_5 \cdot SF_5 \cdots \ln(Size) + \beta_6 \cdot \ln(EM_1) + \ldots + \beta_{22} \cdot \ln(EM_{17}) \quad (7)$$

Multiple regression analysis is performed on the linear model in log space. The derived coefficients, β_i, from regression analysis are used to adjust the *a priori* values. The log-log transformation is supported by analysis of the data which shows other transformations such as log-linear, square root-linear, and quad root-linear resulted in non-constant variance errors. Only the log-log transform satisfies the test for independence between errors and the observations. equation(7) shows the fixed exponent, $Size^{1.01}$, removed from the analysis as we want the scale factors to explain as much of the variance as possible.

In order to calibrate COCOMO II Post-Architecture model, We will estimate coefficients for all cost drivers. There were 83 observations used in the traditional regression techniques and the proposed MRE technique. Of those observations, 59 were used to create a baseline set of coefficients. The response variable were size Person Months (PM). These coefficients are applied to the model using equation (8). The constant, k, is derived from raising e to the coefficient, β_0.

$$PM = e^{\beta_0} \cdot (Size)^{1.01} \cdot (Size)^{\beta_1 \cdot SF_1} \ \ldots \ (Size)^{\beta_5 \cdot SF_5} \cdot EM_1^{\beta_6} \ \ldots \ EM_{17}^{\beta_{22}} \quad (8)$$

To our surprise, some of the coefficients, β_i, were negative in the three calibrating techniques. The negative coefficient estimates do not support the ratings for which the data was gathered. To see the effect of a negative coefficient, <Table 3> gives the ratings, *a priori* values, and fully adjusted values for RUSE which are generated by the LS, RLS, and the proposed MRE calibrating technique. The methods have the negative coefficients for the rating of the required reusability cost driver. The rating for the cost driver RUSE captures the additional effort needed to construct components intended for reuse on the current or future projects. Based on the definition of the Required Reusability cost driver RUSE, the *a priori* model values indicate that as the rating increases from Low (L) to Extra High (XH), the amount of required effort will also increase.

<Table 3>. RUSE Cost Driver

RUSE	Definition	*Adjusted A priori* Values		
		LS Approac	*RLS Approach*	*MRE approach*
L	None	1.05	1.01	1.19
N	Across project	1.00	1.00	1.00
H	Across program	0.94	0.98	1.04
VH	Across product line	0.88	0.91	0.91
XH	Across multiple product lines	0.82	0.85	0.86

The adjusted values determined from the data sample indicate that as more software is built for wider ranging reuse less effort is required. This is inconsistent with the expert-determined multiplier values obtained by the COCOMO II Affiliate representatives. Because it is the low degree of freedom of distribution which is used to calibrate RUSE. There were a lot of responses that were "I don't know" or "It does not apply." There are essentially entered as a Nominal rating in the model. This weakens the data analysis in two way: via weak dispersion of rating values, and via possibly inaccurate data values.

The COCOMO II Affiliate users were reluctant to use a pure regression-based model with counter-intuitive trends for RUSE. We therefore used a weighted average approach to determine an *a posteriori* set of cost driver values as a weighted average of the *a priori* cost drivers and the regression-determined cost drivers. With 10% of the data-driven and 90% of the *a priori* values, <Table 4> shows the COCOMO II calibrated values using the proposed MRE calibrating techniques. The 10% weighting factor was selected after comparison runs using other weighing factors were found to produce less accurate results. This moves the model parameters in the direction suggested by the regression coefficients but retains the rationale contained within the *a priori* values.

<Table 4>. COCOMO II Calibrated values using the MRE method

Driver	Sym	VL	L	N	H	VH	XH
PREC	SF_1	0.0413	0.0374	0.0203	0.0152	0.0078	0.00
FLEX	SF_2	0.0610	0.0468	0.0304	0.0223	0.0121	0.00
RESL	SF_3	0.0422	0.0338	0.0253	0.0169	0.0084	0.00
TEAM	SF_4	0.0488	0.0379	0.0301	0.0201	0.0084	0.00
PMAT	SF_5	0.0444	0.0349	0.0300	0.0162	0.0098	0.00
RELY	EM_1	0.69	0.89	1.00	1.18	1.41	
DATA	EM_2		0.89	1.00	1.11	1.21	
CPLX	EM_3	0.69	0.90	1.00	1.09	1.22	1.71
RUSE	EM_4		0.91	1.00	1.20	1.41	1.61
DOCU	EM_5	0.81	0.94	1.00	1.11	1.21	
TIME	EM_6			1.00	1.19	1.41	1.70
STOR	EM_7			1.00	1.10	1.28	1.59
PVOL	EM_8		0.92	1.00	1.15	1.31	
ACAP	EM_9	1.49	1.20	1.00	0.78	0.62	
PCAP	EM_{10}	1.32	1.10	1.00	0.77	0.69	
PCON	EM_{11}	1.27	1.09	1.00	0.89	0.84	
AEXP	EM_{12}	1.23	1.10	1.00	0.90	0.81	
PEXP	EM_{13}	1.26	1.12	1.00	0.88	0.81	
LTEX	EM_{14}	1.23	1.11	1.00	0.90	0.86	
TOOL	EM_{15}	1.25	1.22	1.00	0.98	0.80	
SITE	EM_{16}	1.24	1.09	1.00	0.90	0.85	0.79
SCED	EM_{17}	1.32	1.20	1.00	1.00	1.00	

As more data is used to calibrate the model, a greater percentage of the weight will be given to the regression determined values. Thus the strategy is to release annual updates to the calibrated parameters with each succeeding update producing more data driven parameter values. The research on the Process Maturity (PMAT) cost driver has shown the data-driven approach to be a credible. A larger dataset was used to determine PMAT's influence on effort. This was caused by the additional data having a wider dispersion of responses for PMAT across its rating criteria.

4.3 Model Accuracy

To evaluate the quality of our proposed calibration MRE technique, some distinct evaluation criteria was chosen as the quantitative measure of goodness for model calibration. First, the calibrated model must adequately represent the dispersion of the data. This is the quality of calibrating criterion. Second, the model must make meaningful future predictions. This is the predictive power of the calibrated model. We consider three criteria which are used to judge how well the model forecasts effort. These criteria are the Adjusted R^2 ($Adj-R^2$), estimated Standard Deviation($\hat{\sigma}$), and Prediction at level L $PRED(L)$. $Adj-R^2$ is the coefficient of determination, R^2, adjusted for the number of cost drivers and the number of observations. Sufficient model consistency as illustrated by the R^2 statistic is an important and desirable aspect of a regression model. $PRED(L)$ is the percentage of estimated values that were within L percent of the actual values. For example, $PRED(0.25)$ reports the percentage of estimates that were within 25% of the actual values.

In order to compare the predictive power of our proposed MRE calibration technique with the traditional regression techniques, Several experiments are have been conducted with all 83 observations. Using 10% data-driven and 90% of the *a priori* values, <Table 5> shows the effort forecast results for the traditional regression techniques and the our proposed MRE calibration technique.

<Table 5>. Effort Forecast Results

Effort Prediction	Calibration methods		
	Traditional LS approach	Traditional RLS approach	Our proposed MRE approach
$Adj-R^2$	0.94	0.94	0.96
$\hat{\sigma}$	1925	1920	1876
$PRED(0.20)$	47%	48%	52%
$PRED(0.25)$	56%	56%	59%
$PRED(0.30)$	66%	67%	70%

A high value for the $Adj-R^2$ indicator is evidence of a strong and consistent linear relationship among two data sets. The $Adj-R^2$ value of our proposed method is higher than that of the other approach. It is evident that the our proposed method have the highest predictive power, as illustrated by the values obtained for $PRED(L)$ measures.

In this table, our proposed calibration method MRE was shown to be superior to the traditional regression approach for model calibration. It may be attributed to the fact that our proposed scheme is insensitive to the outlier for the data source which is a peculiarity and indicates a data point.

5. Concluding Remarks

To date many software engineering cost models have been developed to predict cost, schedule, and effort of the software under development. But, the rapidly changing nature of software development has made it difficult to develop empirical models that continue to yield high prediction accuracies. On the basis of the MRE regression, we proposed a new regression method for calibrating COCOMO II post-architecture model which is well-suited for the new software development life cycle such as non-sequential and rapid-development processes. The characteristic of the proposed method is insensitive to the extreme values of the data in the empirical calibration.

The objective of this study was not to present a definitive calibration method, but rather to improve and evaluate various calibration techniques. As the experimental results, It is evident that our proposed calibration method MRE was shown to be superior to the traditional regression approach for model calibration, as illustrated by the values obtained for standard deviation ($\hat{\sigma}$), and prediction at level L $PRED(L)$ measures.

References

[1] L. H. Putnam and W. Myers, Measures for Excellence, Yourden Press Computing Series, 1992.

[2] C. Jones, Applied Software Measurement, McGraw-Hill, 1997.

[3] B. W. Bohem et al., "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," *Annals of Software Eng. Special Volume on Software Process and Product Measurement*, vol. 1, pp. 45-60, 1995.

[4] F. Walkerden and R. Ross Jeffery, "Software Cost Estimation: A Review of Models, Process and Practices", Advances in Computers, 1997.

[5]. B. Boehm, B. Clark, E. Horowitz, C. Westland. R. Madachy, R. Selby, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," *Annals of Software Engineering Special Volume on Software Process and Product Measurement*, J.D. Arthur and S.M.Henry (Eds.), J.C. Balzer AG, Science Publishers, Amsterdam, The Netherlands, Vol 1, 1995, pp. 45 - 60.

[6] Center for Software Engineering, "COCOMO II Model Definition Manual," Computer Science Department, University of Southern California, Los Angeles, Ca. 90089,
http://sunset .usc.edu/Cocomo.html, 1997

[7] R. Banker, H. Change, and C. Kemerer. "Evidence on Economies of Scale in Software Development," *Information and Software Technology*, vol. 36, no. 5, 1994, pp.275-858.

[8] B. Clark, "The Effects of Process Maturity on Software Development Effort," Ph.D. Dissertation,

Computer Science Department, University of Southern California, Aug. 1997.

[9] IFPUG, Function Point Counting Practices: Manual Release 4.0, International Function Point User's Group, 1994.

[10] R. Selby, "Empirically Analyzing Software Reuse in a Production Environment," in Software Reuse: Emerging Technology, W. Tracz (Ed.), *IEEE Computer Society Press*, 1988, pp.176-189.

[11] T. M. Khoshgftaar, J. C. Munson, B. B. Bhattacharya, and G. D. Richardson, "Predictive Modeling Techniques of Software Quality from Software Measures," *IEEE Trans. on Soft. Eng.*, vol 18, no. 11, pp. 979-987, 1997.

[12] A. Levitin, "The L_1 Criteria in Data Analysis and the Problem of Software Size Estimation," *in Proc. 21st Symp. Interface: Computing Science and Statistics*, pp. 382-384, 1989.

[13] W. Griffiths, R. Hill, and G. Judge, Learning and Practicing Econometrics, John Wiley & Sons, Inc., New York, N.Y., 1993.

[14] R. Park, "Software Size Measurement: A Framework for Counting Source Statements," CMU/SEI-92-TR-20, Software Engineering Institute, Pittsburgh, Pa., 1992.

[15] S. Weisberg, Applied Linear Regression, 2nd Ed., John Wiley and Sons, New York, N.Y., 1985.

윤 명 영

1984년 충북대학교 계산통계학과 졸업 (학사)
1987년 충북대학교 대학원 계산통계학과(이학석사)
1997년 충북대학교 대학원 전자계산학과(이학박사)
1987년~1990년 육군 전산장교 근무
1992년~현재 충청대학 컴퓨터학부 부교수
관심분야 : 화상처리, 패턴인식, 멀티미디어