Calibrating the COCOMO II Post-Architecture Model

Bradford Clark
Center for Software Engg.
Computer Science Department
Univ. of Southern California
Los Angeles, CA 90089, USA
+1 760 939 8279
bkclark@sunset.usc.edu

Sunita Devnani-Chulani
Center for Software Engg.
Computer Science Department
Univ. of Southern California
Los Angeles, CA 90089, USA
+1 213 740 6470
sdevnani@sunset.usc.edu

Barry Boehm
Center for Software Engg.
Computer Science Department
Univ. of Southern California
Los Angeles, CA 90089, USA
+1 213 740 5703
boehm@sunset.usc.edu

ABSTRACT
The COCOMO II model was created to meet the need for a cost model that accounted for future software development practices. This paper describes some of the experiences learned in calibrating COCOMO II Post-Architecture model from eighty-three observations. The results of the multiple regression analysis, their implications, and a future calibration strategy are discussed.

Keywords
COCOMO, cost estimation, metrics, multiple regression.

1 INTRODUCTION
The COCOMO II project started in July of 1994 with the intent to meet the projected need for a cost model that would be useful for the next generation of software development. The new model incorporated proven features of COCOMO 81 and Ada COCOMO models. COCOMO II has three submodels. The Application Composition model is used to estimate effort and schedule on projects that use Integrated Computer Aided Software Engineering tools for rapid application development. The Early Design and Post-Architecture models are used in estimating effort and schedule on software infrastructure, major applications, and embedded software 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. Compared to COCOMO 81, COCOMO II added new cost drivers for application precedentedness, development flexibility, architecture and risk resolution, team cohesion, process maturity, required software reuse, documentation match to lifecycle needs, personnel continuity, and multi-site development. COCOMO II also eliminated COCOMO 81's concept of development modes and two COCOMO 81 cost drivers: turnaround time and modern programming practices.

This paper describes our experiences and results of the first calibration of the Post-Architecture model. The model determination process began with an expert Delphi process to determine apriori values for the Post-Architecture model parameters. The dataset of 83 projects was used for model calibration. Model parameters that exhibited high correlation were consolidated. Multiple regression analysis was used to produce coefficients. These coefficients were used to adjust the previously assigned expert-determined model values. Stratification was used to improve model accuracy.

The resulting model produces estimates within 30% of the actuals 52% of the time for effort. If the model's multiplicative coefficient is calibrated to each of the major sources of project data, the resulting model produces estimates within 30% of the actuals 64% of the time for effort. It is therefore recommended that organizations using the model calibrate it using their own data. This increases model accuracy and produces a local optimum estimate for similar type projects.

Section 2 of this paper reviews the structure of the COCOMO II Post-Architecture model. Section 3 describes the data used for calibration. Section 4 describes the calibration procedures, results, and future calibration strategy.

2 POST-ARCHITECTURE MODEL
The COCOMO II Post-Architecture model is fully described in [1&3]. The Post-Architecture model covers the actual development and maintenance of a software product. This stage of the lifecycle 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. 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), 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. The model has the form:

$$PM = A \cdot (Size)^{1.01+\sum_{j=1}^{5} SF_j} \cdot \prod_{i=1}^{17} EM_i \qquad (1)$$

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.

Recent research has confirmed diseconomies of scale influence on effort with the Process Maturity (PMAT) scaling cost driver [2]. For projects in the 30 - 120 KSLOC range, the analysis indicated that a one level improvement in Process Maturity corresponded to a 15 - 21% reduction in effort, after normalization for the effects of other cost drivers.

Table 1. COCOMO II Cost Drivers

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

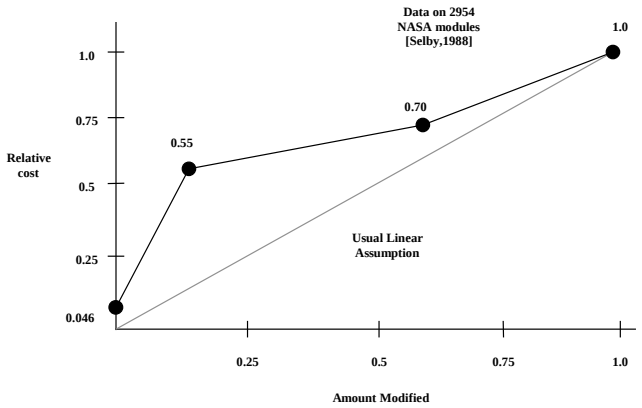
All of the cost drivers are listed in Table 1. Each driver can accept one of six possible qualitative ratings: Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH), and Extra High (XH). Not all ratings are valid for all cost drivers. An example of ratings for a cost driver is given for the Required Reusability (RUSE) cost driver, see Table 2.

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) or thousands of source lines of code (KSLOC).

The COCOMO II size reuse model is nonlinear and is based on research done by Selby [5]. Selby's analysis of reuse costs across nearly 3000 reused modules in the NASA Software Engineering Laboratory indicates that the reuse cost function is nonlinear in two significant ways (see Figure 1):

- It does not go through the origin. There is generally a cost of about 5% for assessing, selecting, and assimilating the reusable component.
- Small modifications generate disproportionately large costs. This is primarily due to two factors: the cost of understanding the software to be modified, and the relative cost of interface checking.

Figure 1. Non-linear Effects of Reuse



The COCOMO II sizing model captures this non-linear effect with six parameters: 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) for structure, clarity, and self-descriptiveness; unfamiliarity with the software (UNFM) for programmer knowledge of the reused code; and assessment and assimilation (AA) for fit of the reused module to the application [3].

3 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. These 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. 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.

A frequent question is what defines a line of source code. Appendix B in the Model Definition Manual [3] defines a logical line of code. However the data collected to date has exhibited local variations in interpretation of counting rules, which is one of the reasons that local calibration produces more accurate model results.

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.

Figure 2. Data Distribution for Person Months

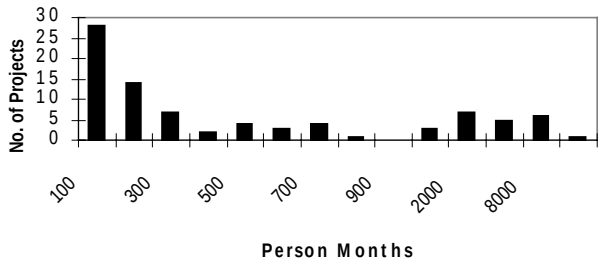
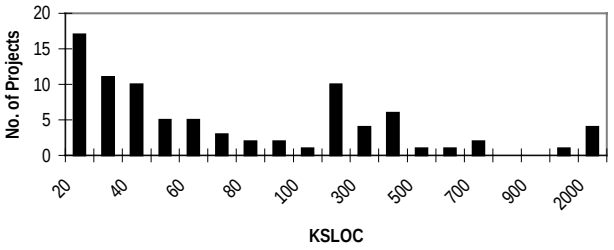


Figure 3. Data Distribution for Size



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.

We have found that the different definitions of product and development in the 1990's make data sources less comparable than in the 1970's. As a result, even after data normalization the accuracy of COCOMO II.1997 was less accurate on 83 projects than COCOMO 1981 was on 63 projects.

4. MODEL CALIBRATION

The statistical method we used to calibrate the model is called multiple regression analysis. This analysis finds the least squares error solution between the model parameters and the actual effort, PM, expended on the project. The COCOMO model as shown in Equation 1 is a non-linear model. To solve this problem we transform the non-linear model in Equation 1 into a linear model using logarithms to the base *e*, Equation 2.

$$\ln(PM) = \ln A + 1.01\ln(Size) + SF_1 \ln(Size) + \Lambda + SF_5 \ln(Size) + \ln(EM_1) + \Lambda + \ln(EM_{17}) \quad (2)$$

The next step was to heuristically set the values of the exponential and multiplicative qualitative cost drivers. This was done using a Delphi process with the COCOMO II Affiliate users. These are called apriori values.

Multiple regression analysis was performed on the linear model in log space. The derived coefficients, B_i, from regression analysis were used to adjust the apriori values.

4.1 Results of Effort Calibration

There were 83 observations used in the multiple regression analysis. Of those observations, 59 were used to create a baseline set of coefficients. The response variable was Person Months (PM). The predictor variables were size (adjusted for reuse and breakage) and all of the cost drivers listed in Table 1. The constant, A, is derived from raising *e* to the coefficient, B_0, Equation 2.

To our surprise, some of the coefficients, B_i, were negative. The negative coefficient estimates do not support the ratings for which the data was gathered. To see the effect of a negative coefficient, Table 2 gives the ratings, apriori values, and fully calibrated values for RUSE. The rating for the Required Reusability cost driver, RUSE, captures the additional effort needed to construct components intended for reuse on the current or future projects. The apriori model values indicate that as the rating increases from Low (L) to Extra High (XH), the amount of required effort will also increase. This rationale is consistent with the results of 12 studies of the relative cost of writing for reuse compiled in [4]. The adjusted values determined from the data sample indicate that as more software is built for wider ranging reuse less effort is required. As shown in Figure 4, diamonds versus triangles, this is inconsistent with the expert-determined multiplier values obtained via the COCOMO II Affiliate representatives.

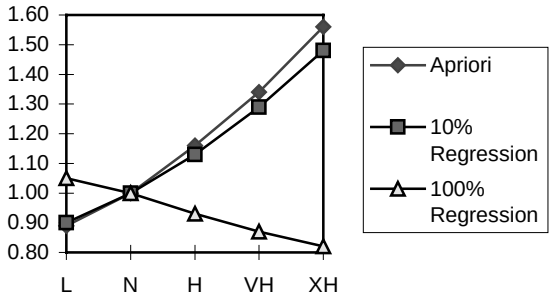
A possible explanation for the phenomenon is the frequency distribution of the data used to calibrate RUSE. There were a lot of responses that were "I don't know" or "It does not apply." These are essentially entered as a Nominal rating in the model. This weakens the data analysis in two ways: via weak dispersion of rating values, and via possibly inaccurate data values. The regression analysis indicated that variables with negative were not statistically significant for this dataset. This appears to be due to lack of dispersion for some variables; imprecision of

software effort, schedule, and cost driver data; and effects of partially correlated variables.

Table 2. RUSE Cost Driver

RUSE	Definition	Values	
		Apriori	Adjusted
L	None	0.89	1.05
N	Across project	1.00	1.00
H	Across program	1.16	0.94
VH	Across product line	1.34	0.88
XH	Across multiple product lines	1.56	0.82

Figure 4. RUSE Calibrated Values



4.2 Strategy and Future Calibration

We and the COCOMO II affiliate users were reluctant to use a pure regression-based set of cost driver values which conflicted with the expert-determined apriori values such as the triangles in Figure 4 for RUSE. We therefore used a 10% weighted-average approach to determine an aposteriori set of cost driver values as a weighted average of the apriori cost drivers and the regression-determined cost drivers. Using 10% of the data-driven and 90% of the apriori values.

The 10% weighting factor was selected after comparison runs using 0% and 25% weighing factors were found to produce less accurate results than the 10% factors. This moves the model parameters in the direction suggested by the regression coefficients but retains the rationale contained within the apriori values. 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. Hence the COCOMO II model name will have a year date after it identifying the set of values on which the model is based, e.g. COCOMO II.1997.

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. PMAT was statistically significant with the large 112 project dataset compared to the 83 project dataset discussed here[2]. This was caused by the additional data having a wider dispersion of responses for PMAT across its rating criteria.

Stratification produced very good results. As with the previous COCOMO models, calibration of the constant, A, and the fixed exponent, 1.01, to local conditions is highly recommended. This feature is available in a commercial implementation of COCOMO II, COSTAR's Calico tool, and is also available in the free software tool, USC COCOMO II.1997.1.

4.3 Future Work

With more data we are going to make a thorough examination of the cost drivers that have a negative coefficient. We plan to extend the Bayesian approach to address cost drivers individually. There is also a need to calibrate cost drivers for their influence during maintenance. The distribution of effort across lifecycle phases needs to be updated. This is challenging because of different lifecycle models that are used today, e.g. spiral, iterative, evolutionary. COCOMO II's sizing model is very comprehensive but there was not enough data to fully check its validity. Additionally the relationship between Unadjusted Function Points and logical / physical source lines of code needs further study.

REFERENCES

1. Boehm, B., 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. Baltzer AG, Science Publishers, Amsterdam, The Netherlands, Vol 1, 1995, pp. 45 - 60.
2. Clark, B., "The Effects of Process Maturity on Software Development Effort," Ph.D. Dissertation, Computer Science Department, University of Southern California, Aug. 1997.
3. 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.
4. Poulin, J., "Measuring Software Reuse", Addison-Wesley, Reading, Ma., 1997.
5. Selby, R., "Empirically Analyzing Software Reuse in a Production Environment," in Software Reuse: Emerging Technology, W. Tracz (Ed.), IEEE Computer Society Press, 1988, pp.176-189.
6. Weisberg, S., Applied Linear Regression, 2nd Ed., John Wiley