Calibrating the COCOMO II Post-Architecture Model

Sunita Devnani-Chulani
Bradford Clark
Barry Boehm
Center for Software Engineering
Computer Science Department
University of Southern California
Los Angeles, CA 90098-0781 USA
+1 213 740 6470
sdevnani@sunset.usc.edu
bkclark@sunset.usc.edu
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 resulted in the formulation of three submodels for cost estimation, one for composing applications, one for early lifecycle estimation and one for detailed estimation when the architecture of the product is understood. This paper describes the calibration procedures for the last model, Post-Architecture COCOMO II model, from eighty-three observations. The results of the multiple regression analysis and their implications are discussed. Future work includes further analysis of the Post-Architecture model, calibration of the other models, derivation of maintenance parameters, and refining the effort distribution for the model output.

Keywords
COCOMO II, 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 the calibration of the Post-Architecture model. The model determination process began with an expert Delphi process to determine a-priori values for the Post-Architecture model parameters. 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. 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 t-values from the statistical analysis indicated that several of the variables' coefficients were not statistically significant. 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. As a result, the operational COCOMO II.1997 model uses a weighted average approach to determine an a-posteriori model as a percentage of the expert-determined a-priori cost-driver values and the data-determined values.

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 discusses the COCOMO II model that was calibrated. Section 3 describes the data used for calibration. Section 4 describes the calibration procedures, results, and future calibration strategy. Section 5 discusses the model forecast accuracy. Section 6 has our conclusions.

2 POST-ARCHITECTURE MODEL

The COCOMO II Post-Architecture model is fully described in [3 and 6]. 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. 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). 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:

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

All of the cost drivers are described in Table 1. 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). Not all ratings are valid for all cost drivers. Table 2 shows the apriori values assigned to each rating before calibration.

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 first five cost drivers in Table 1 are the five model scale factors. Each scale driver has a range of rating levels, from Very Low to Extra High. Each rating level has a specific value as shown in Table 2. A project's scale factors, $SF_{1\text{-}5}$, are summed and used to determine a scale exponent. 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. Larger projects will have more personnel, and thus more interpersonal communications paths consuming overhead. Integrating a small product as part of a larger product requires not only the effort to develop the small product, but also the additional overhead effort to design, maintain, integrate, and test its interfaces with the remainder of the product. See [1] for a further discussion of software economies and diseconomies of scale.

Recent research has confirmed diseconomies of scale influence on effort with the Process Maturity (PMAT) scaling cost driver [4]. An econometric log-log model and the COCOMO II Post-Architecture model were used to determine effect on effort. The log-log model employed PMAT as a multiplicative cost driver and the COCOMO II Post-Architecture model used PMAT as shown in Equation 1. The multiple regression results showed that PMAT was more significant when used as a scaling cost driver. 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

Abr.	Name and Description
PREC	Precendentedness: Captures the organization's understanding of product objectives and required technology.
FLEX	Development Flexibility: Expresses the degree of conformance to software requirements and external interface standards.
RESL	Architecture and Risk Resolution: Rates the integrity and understanding of the product software architecture and the number / criticality of unresolved risk items.
TEAM	Team cohesion: Captures the consistency of stakeholder objectives and the willingness of all parties to work together as a team.
PMAT	Process Maturity: Maturity of the software process used to produce the product; based on the Software Capability Maturity Model

Abr.	Name and Description
RELY	Required Software Reliability: Degree of assurance to which the software will perform its intended function.
DATA	Data Base Size: Captures the effect that large data requirements have on product development.
CPLX	Product Complexity: Characterizes the product's complexity in five areas; control operations, computational operations, device-dependent operations, data management operations, and user interface management operations
RUSE	Required Reusability: Accounts for the additional effort needed to construct components intended for reuse on the current or future projects.
DOCU	Documentation Match to Life-cycle Needs: Evaluated in terms of the suitability of the project's documentation to its life-cycle needs
TIME	Time Constraint: Measure of the execution time constraint imposed upon a software system.
STOR	Storage Constraint: Degree of main storage constraint imposed on a software system.
PVOL	Platform Volatility: Volatility of the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks.
ACAP	Analyst Capability: Captures analysis and design ability, efficiency and thoroughness, and the ability to communicate and cooperate.
PCAP	Programmer Capability: Captures ability, efficiency and thoroughness, and the ability to communicate and cooperate. Considers programmers as a team rather than as individuals.
AEXP	Applications Experience: Rates the level of applications experience of the project team.
PEXP	Platform Experience: Experience in using the complex of hardware and software (OS, DBMS, etc.) the software product calls on to perform its tasks.
LTEX	Language and Tool Experience: Measure of the level of programming language and software tool experience of the project team.
PCON	Personnel Continuity: Project's annual personnel turnover.
TOOL	Use of Software Tools: Captures the productivity impact of tools ranging from simple edit and code tools to integrated, pro-active lifecycle support tools.
SITE	Multi-Site Development: Assesses site collocation (from fully collocated to international distribution) and communication support (from surface mail and some phone access to full interactive multimedia).
SCED	Required Development Schedule: Measures the schedule constraint imposed on the project team

Table 2 shows the apriori values assigned to the scaling and multiplicative cost drivers. An example of the criteria for selecting a rating for a driver is given for the Required Reusability (RUSE) cost driver in Table 4. 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 (see Table 4).

Table 2. Apriori Model Values

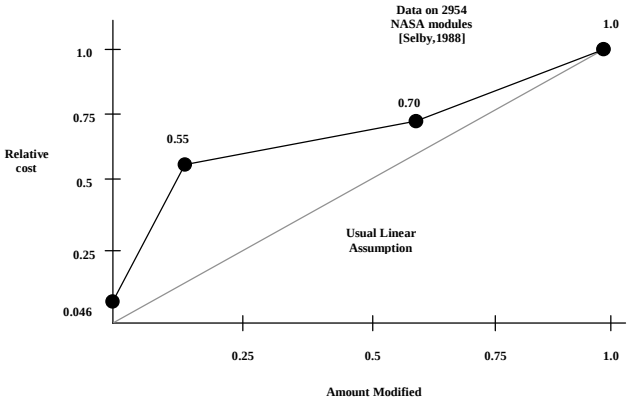
Driver	Sym	VL	L	N	H	VH	XH
PREC	SF_1	0.05	0.04	0.03	0.02	0.01	0.0
FLEX	SF_2	0.05	0.04	0.03	0.02	0.01	0.0
RESL	SF_3	0.05	0.04	0.03	0.02	0.01	0.0
TEAM	SF_4	0.05	0.04	0.03	0.02	0.01	0.0
PMAT	SF_5	0.05	0.04	0.03	0.02	0.01	0.0
RELY	EM_1	0.75	0.88	1.00	1.15	1.40	
DATA	EM_2		0.94	1.00	1.08	1.16	
CPLX	EM_3	0.75	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 [12]. 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 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 [7]. 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 a different location to protect 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 [6] defines a logical line of code using a framework described in [10]. 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. 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.

Figure 2. Data Distribution for Person Months

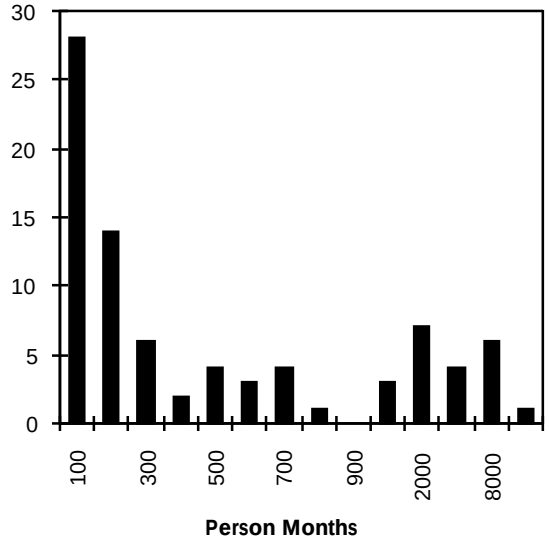


Figure 3. Data Distribution for Schedule

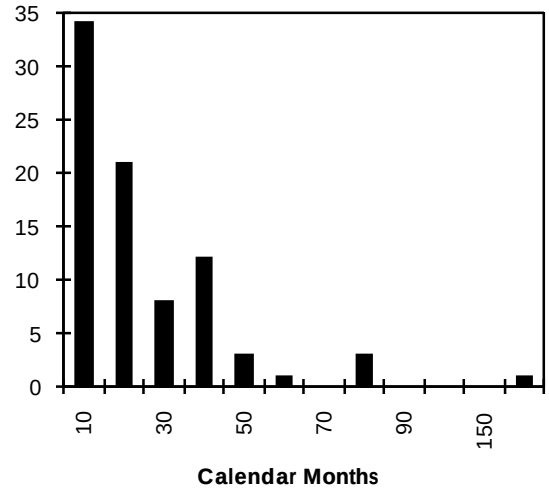
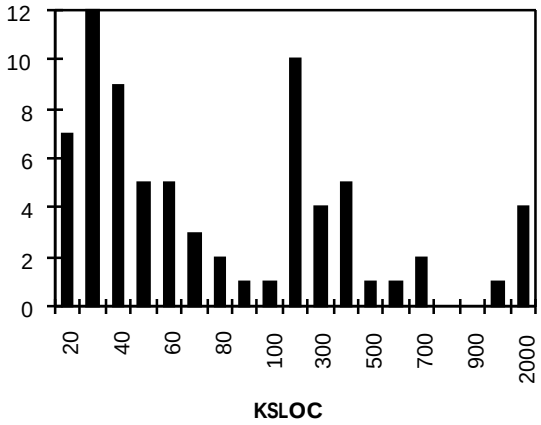


Figure 4. Data Distribution for Size



4. MODEL CALIBRATION

The comprehensive calibration method adjusts all cost driver parameters in the Post-Architecture model. This level of analysis requires a lot of data; usually more than an organization has available. We plan to annually update the model parameter values based on newly added and existing data in our repository. 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 statistical method we used to calibrate the model is multiple regression analysis. This analysis finds the least squares error solution between the model parameters and the data. However, multiple regression analysis, which derives the values for b_i, can only be applied to linear models: $b_0 + b_1A + b_2B + b_3C = D$. The COCOMO model as shown in Equation 2 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 Equation 2 into a linear model with three steps.

The first step is to expand Equation 1 into unique factors. Each factor will be calibrated.

$$PM = A \cdot (Size)^{1.01} \cdot (Size)^{SF_1} \cdot \cdots \cdot (Size)^{SF_5} \cdot EM_1 \cdot EM_2 \cdots EM_{16} \cdot EM_{17} \quad (2)$$

The second step is to heuristically set the values of the exponential and multiplicative cost drivers. These values are the apriori values shown earlier in Table 2. The third step is to transform both sides of the multiplicative form of Equation 2 into a linear form using logarithms to the base *e*. This technique is called a log-log transformation [13].

$$\ln(PM \div Size^{1.01}) = B_0 + B_1 \cdot SF_1 \cdot \ln(Size) + B_2 \cdot SF_2 \cdot \ln(Size) + \cdots + B_5 \cdot SF_5 \cdot \ln(Size) + B_6 \cdot \ln(EM_1) + B_7 \cdot \ln(EM_2) + \cdots + B_{21} \cdot \ln(EM_{16}) + B_{22} \cdot \ln(EM_{17}) \quad (3)$$

Multiple regression analysis is performed on the linear model in log space. The derived coefficients, B_i, from regression analysis are used to adjust the apriori 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 3 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.

For calibration purposes some cost drivers in the original definition of the model were aggregated into new cost drivers because of the high correlation between them. These high correlations introduce numerical instabilities into the data analysis. The Analyst Capability and Programmer Capability were combined to form the Personnel Capability (PERS) cost driver. Execution Time Constraint and Main Storage Constraint were combined to form Resource Constraints (RCON): EM_{14} and EM_{15} in Table 3.

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 described in Table 1. The coefficients obtained from the analysis are shown in Table 3. These coefficients are applied to the model using Equation 4. The constant, A, is derived from raising *e* to the coefficient, B_0.

$$PM = e^{B_0} \cdot (Size)^{1.01} \cdot (Size)^{B_1 \cdot SF_1} \cdot (Size)^{B_2 \cdot SF_2} \cdot \ldots \cdot (Size)^{B_5 \cdot SF_5} \cdot EM_1^{B_6} \cdot EM_2^{B_7} \cdot \ldots \cdot EM_{14}^{B_{19}} \cdot EM_{15}^{B_{20}} \quad (4)$$

Table 3. Estimated Coefficients

Sym	Coefficient	Estimate	t-value
A	B_0	0.70188	3.026
PREC	B_1 for SF_1	-0.90196	-0.621
FLEX	B_2 for SF_2	3.14218	2.074
RESL	B_3 for SF_3	-0.55861	-0.293
TEAM	B_4 for SF_4	0.86614	0.509
PMAT	B_5 for SF_5	0.08844	0.068
RELY	B_6 for EM_1	0.79881	1.511
DATA	B_7 for EM_2	2.52797	3.493
RUSE	B_8 for EM_3	-0.44410	-0.913
DOCU	B_9 for EM_4	-1.32819	-1.999
CPLX	B_{10} for EM_5	1.13191	2.605
PVOL	B_{11} for EM_6	0.85830	1.612
AEXP	B_{12} for EM_7	0.56053	0.920
PEXP	B_{13} for EM_8	0.69690	1.321
LTEX	B_{14} for EM_9	-0.04214	-0.063
PCON	B_{15} for EM_{10}	0.30826	0.260
TOOL	B_{16} for EM_{11}	2.49512	2.243
SITE	B_{17} for EM_{12}	1.39701	1.679
SCED	B_{18} for EM_{13}	2.84075	3.670
PERS	B_{19} for EM_{14}	0.98747	4.282
RCON	B_{20} for EM_{15}	1.36588	5.001

The negative coefficient estimates do not support the ratings for which the data was gathered. To see the effect of a negative coefficient, Table 4 gives the ratings, apriori values, and fully adjusted values for RUSE. Based on the definition of the Required Reusability cost driver, RUSE, 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 [11]. 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 5, diamonds versus triangles, this is inconsistent with the expert-determined multiplier values obtained via the COCOMO II Affiliate representatives.

Table 4. 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

A possible explanation for the phenomenon is the frequency distribution of the data used to calibrate RUSE, Figure 6. 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.

Figure 5. RUSE Calibrated Values

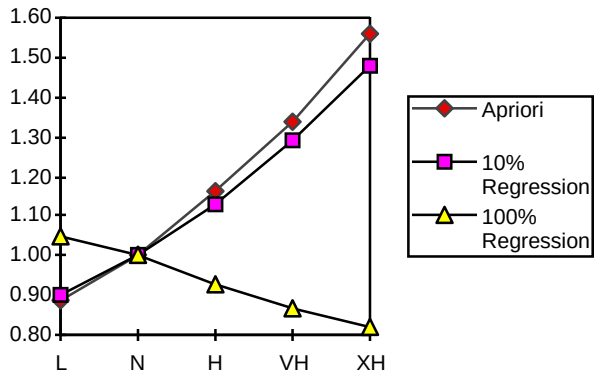
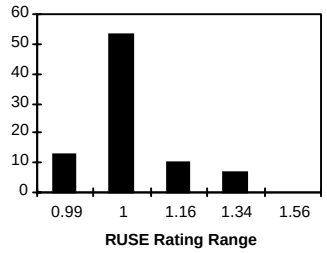


Figure 6. Frequency of RUSE



The COCOMO II Affiliate users were reluctant to use a pure regression-based model with counterintuitive trends such as the triangles in Figure 5 for RUSE. We therefore used a 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, Table 5 shows the COCOMO II.1997 calibrated values. The constant, A, is set to 2.45.

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 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.

Table 5. COCOMO II.1997 Cost Driver Values

Driver	Sym	VL	L	N	H	VH	XH
PREC	SF_1	0.0405	0.0324	0.0243	0.0162	0.0081	0.00
FLEX	SF_2	0.0607	0.0486	0.0364	0.0243	0.0121	0.00
RESL	SF_3	0.0422	0.0338	0.0253	0.0169	0.0084	0.00
TEAM	SF_4	0.0494	0.0395	0.0297	0.0198	0.0099	0.00
PMAT	SF_5	0.0454	0.0364	0.0273	0.0182	0.0091	0.00
RELY	EM_1	0.75	0.88	1.00	1.15	1.39	
DATA	EM_2		0.93	1.00	1.09	1.19	
CPLX	EM_3	0.75	0.88	1.00	1.15	1.30	1.66
RUSE	EM_4		0.91	1.00	1.14	1.29	1.49
DOCU	EM_5	0.89	0.95	1.00	1.06	1.13	
TIME	EM_6			1.00	1.11	1.31	1.67
STOR	EM_7			1.00	1.06	1.21	1.57
PVOL	EM_8		0.87	1.00	1.15	1.30	
ACAP	EM_9	1.50	1.22	1.00	0.83	0.67	
PCAP	EM_{10}	1.37	1.16	1.00	0.87	0.74	
PCON	EM_{11}	1.24	1.10	1.00	0.92	0.84	
AEXP	EM_{12}	1.22	1.10	1.00	0.89	0.81	
PEXP	EM_{13}	1.25	1.12	1.00	0.88	0.81	
LTEX	EM_{14}	1.22	1.10	1.00	0.91	0.84	
TOOL	EM_{15}	1.24	1.12	1.00	0.86	0.72	
SITE	EM_{16}	1.25	1.10	1.00	0.92	0.84	0.78
SCED	EM_{17}	1.29	1.10	1.00	1.00	1.00	

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 dataset compared to the unknown significance shown in Table 3 (the t-value is less than 1.96) [4]. This was caused by the additional data having a wider dispersion of responses for PMAT across its rating criteria.

4.2 Results of Schedule Calibration

COCOMO II provides an estimation of schedule (TDEV), Equation 5. Only the constant for the schedule equation was calibrated. The calibrated schedule constant, A, is set to 2.66. The apriori value was 3.0.

$$TDEV = [A \cdot (\overline{PM})^{(0.33+0.2\cdot\sum SF_i)}] \cdot \frac{SCED\%}{100} \quad (5)$$

$\overline{PM}$ is the estimation of effort without the *SCED* cost driver included. *SF* are the scaling cost drivers earlier.

5 MODEL ACCURACY

5.1 Criteria

Four criteria are used to judge how well the model forecasts effort and schedule. These criteria are the Adjusted R^2 (Adj-R^2), estimated Standard Deviation ($\hat{\sigma}$), and Prediction at level L (PRED(L)). Adjusted R^2 is the coefficient of determination, R^2, adjusted for the number of cost drivers

and the number of observations [8]. R^2 reports the proportion of variation explained by the model.

The estimated Standard Deviation is an indication of the variation in the error of the prediction. It is based on the difference between the known effort to complete a software project and the model estimated effort.

PRED(L) is the percentage of estimated values that were within L percent of the actual values [5]. For example, PRED(.25) reports the percentage of estimates that were within 25% of the actual values. This measure depends on the measurement of error. The error measurement used in our results is called Proportional Error.

Proportional Error (PE) is a measure of relative error used to compute PRED(L). As the estimates become larger for larger projects, the difference between the known effort to complete a software project and the model estimated effort is normalized for project size. Equation 6 shows this normalization.

$$PE = \begin{cases} [PM_{est} \div PM_{act}] - 1, (PM_{est} - PM_{act}) \geq 0 \\ -[PM_{act} \div PM_{est}] + 1, (PM_{est} - PM_{act}) < 0 \end{cases} \quad (6)$$

5.2 Results

With all 83 observations, the 10% adjusted values for the COCOMO II parameters produced the effort forecast measurements shown in Table 6. There are two columns, Before Stratification and After Stratification. The first column shows accuracy results using the same constant, A, and cost drivers for all observations. The second column shows the accuracy results using a different constant, A, and different fixed exponent for each different organization that contributed data. The results are better for the stratified model. The Standard Deviation ($\hat{\sigma}$) for the actual effort values was 1953 Person Months. It can be seen that the $\hat{\sigma}$ for the estimated values in the stratified model is better than the $\hat{\sigma}$ for the actual values.

Table 6. Effort Forecast Results

Effort Prediction	Before Stratification	After Stratification
Adj_R^2	0.93	0.95
$\hat{\sigma}$	2165	1925
PRED(.20)	42%	47%
PRED(.25)	47%	56%
PRED(.30)	51%	66%

These results are not as good as the COCOMO 81 model. This we believe is due to the small amount of data available for model calibration. However, it can be seen that calibration of the model to local conditions increases accuracy.

6 CONCLUSIONS

Data collection is still on going. The first calibrated release of the model occurred early in 1997. Until we can produce 100% data driven values, COCOMO II will under go annual updates to all of its cost drivers using the methods described in this paper. Each release will increasingly depend on data-driven values.

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.

In our data collection, there is now an entry for "I don't know." This reflects the paradox in attempting to create a cost model for the future using data from the past. Cost drivers such as RUSE reflect practices that were not emphasized in historical project data.

Future Work

With more data we are going to make a through examination of the cost drivers that have a negative coefficient. The theory upon which the model is based does not support negative coefficients. Our experience with the Process Maturity cost driver has demonstrated the positive effects of using a larger dataset. There is also a need to calibrate cost drivers for their influence during maintenance. COCOMO 81 had this feature. 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.

ACKNOWLEDGMENTS

Data, expertise, and funding support from the 27 COCOMO II Affiliates: Aerospace Corporation, Air Force Cost Analysis Agency, Allied Signal, Bellcore, The Boeing Co., E-Systems/Raytheon, Electronic Data Systems Corporation, Federal Aviation Administration, Hughes Aircraft Company, Institute for Defense Analysis, Jet Propulsion Laboratory, Litton Data Systems, Lockheed/Martin, Lucent Technologies, Motorola Inc., Network Programs, Inc., Northrop Grumman Corporation, Rational Software Corporation, Science Applications International Corporation, Software Engineering Institute (CMU), Software Productivity Consortium, Sun Microsystems Inc., Texas Instruments, TRW, Inc., U.S. Air Force Rome Laboratory, U.S. Army Research Laboratory, and Xerox Corporation.

REFERENCES

1. Banker, R., H. Change, and C. Kemerer, "Evidence on Economies of Scale in Software Development," Information and Software Technology, vol. 36, no. 5, 1994, pp.275-828.
2. Boehm, B., Software Engineering Economics, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1981.
3. 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.
4. Clark, B., "The Effects of Process Maturity on Software Development Effort," Ph.D. Dissertation, Computer Science Department, University of Southern California, Aug. 1997.
5. Conte, S., H. Dunsmore, and V. Shen, Software Engineering Metrics and Models, Benjamin/Cummings, Menlo Park, Ca., 1986.
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. Center for Software Engineering, "COCOMO II Cost Estimation Questionnaire," Computer Science Department, University of Southern California, Los Angeles, Ca. 90089, http://sunset.usc.edu/Cocomo.html, 1997.
8. Griffiths, W., R. Hill, and G. Judge, Learning and Practicing Econometrics, John Wiley & Sons, Inc., New York, N.Y., 1993.
9. IFPUG, Function Point Counting Practices: Manual Release 4.0, International Function Point User's Group, 1994.
10. Park. R., "Software Size Measurement: A Framework for Counting Source Statements," CMU/SEI-92-TR-20, Software Engineering Institute, Pittsburgh, Pa., 1992.
11. Poulin, J., Measuring Software Reuse, Addison-Wesley, Reading, Ma., 1997.
12. 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.
13. Weisberg, S., Applied Linear Regression, 2nd Ed., John Wiley and Sons, New York, N.Y., 1985.

22nd SEW COVER PAGE

TITLE OF PAPER:

Calibrating the COCOMOII post Architecture Model

CONTACT PERSON:

Name: Sunita Devnani Chulani

Affiliation : Center for Software Engineering, University Of Southern California

Street : Dept Of Computer Science

Address: University Of Southern California

City : Los Angeles

State/Province : CA

Postal/Zip Code : 90089 –0781

Country: US

Email: sdevnani@sunset.usc.edu

Telephone : 213 740 4927

Additional Authors

Name : Brad Clark

Affiliation : Center for Software Engineering, University Of Southern California

Name : Barry Boehm

Affiliation : Center for Software Engineering, University Of Southern California