International Journal of Information Technology and Business Management

29th November 2012. Vol.7 No.1

ISSN 2304-0777 www.jitbm.com

Assessing Accuracy of Formal Estimation Models and Development of an Effort Estimation Model for Industry Use

Paula S. Esplanada
Department of Computer Science,University of the Philippines Cebu College
Lahug, Cebu City 6000, Philippines

Eliezer A. Albacea
Institute of Computer Science, University of the Philippines Los Baños
College, Laguna 4031, Philippines

Email: paula.esplanada@gmail.com eaa@ics.uplb.edu.ph,

Abstract

The need for accurate estimates in software development continues to grow as more and more companies are experiencing cost overruns due to poor estimates. This study evaluates four of the most popular algorithmic models used to estimate software effort (Basic COCOMO, Intermediate COCOMO, Walston-Felix, and Bailey-Basili) using project data from a particular software company. The results show that these models cannot be used for the organization and should instead only be used as a rough estimate; thus, an estimation model is also developed for the company. Tests showed that the model developed is capable of estimating effort with a certain degree of accuracy although the model may be further refined by accounting and quantifying other factors that affect productivity. More focus should be done in creating a model and accounting cost factors for web application project types. Other size metrics apart from LOC should also be considered.

Keywords; Software cost models, lines of code, effort estimation, metrics-performance measures, cost estimation.

I. INTRODUCTION

Estimating the cost of a software product is one of the most complex and error-prone tasks. It is difficult to make an accurate cost estimate during the planning phase of software development because of many unidentified factors at that stage. On the other hand, a solid cost commitment is often needed as part of the feasibility study. Along with the competitive nature of this business, this is a major factor that contributes to the widespread cost and schedule overruns of software projects. Because of this, many project managers struggle with estimating projects [1]. A good cost estimate early in the project's life helps the project managers know how many developers will be required and enables them to arrange for the necessary staff to be available when needed.

To address the need for producing accurate estimates, certain techniques have been developed to capture the relationship among effort and staff characteristics, project requirements and other factors that can affect the time, effort and cost of developing a software system [2]. There are now a lot of algorithmic models that can help compute for and estimate the total cost and development time of a software development project, based on estimates of a restricted number of significant cost drivers [3].

II. BACKGROUND

A. Basic COCOMO

The COCOMO cost estimation model (1981) is used by thousands of software project managers, and is based on a study of hundreds of software projects. This is an acronym derived from the first two letters of each word in the longer phrase COnstructive COst Model [4]. The most fundamental calculation in the COCOMO model is

29th November 2012. Vol.7 No.1

ISSN 2304-0777 www.jitbm.com

the use of the Effort Equation to estimate the number of Person-Months required in developing a project. Most of the other COCOMO results, including the estimates for Requirements and Maintenance, are derived from this quantity [5]. The initial version of COCOMO (COCOMO 81) was developed by Dr. Barry Boehm and published in Software Engineering Economics [6] in 1981. It uses a basic regression formula to predict effort, schedule, cost and staffing of software projects based on historical data.

Boehm analyzed a set of 63 software development projects to find out what influences the effort and development time in software projects. He identified the size of the system to be the main influence factor on the effort. Further influence factors were the type of development project, the developers' skills as well as performance characteristics. There are formulas to convert the person month estimates into schedule estimates and cost. The basic formula of the basic COCOMO model is:

$$EFFOFT\ (PM) = a * (KDSI)^{b} \quad (1)$$

where KDSI is a measure to determine the software size, namely the Kilo Delivered Source Instruction (or Kilo Lines of Code). The constants *a* and *b* depend on the type of development mode. For the projects where the actual effort and cost were available, the influence factors were determined. The constant *a* has a linear affect on the effort prediction, whereas *b* is an exponential factor. *b* is used to account for diseconomies of scale: The effort to develop a system increases exponentially with the size of the system. One reason for this is the increased communication and management activities [7].

Table 1. COCOMO Basic Model Effort Equations [7]

Development mode	Effort Equation
Organic	EFFORT (PM) = 2.4 * (KDSI) 1.05
Semi-detached	EFFORT (PM) = 3.0 * (KDSI) 1.12
Embedded	EFFORT (PM) = 3.6 * (KDSI) 1.20

Organic projects are relatively small, simple software projects in which small teams with good application experience work to a set of less than rigid requirements. Semi-detached projects are intermediate (in size and complexity) software project in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements. Embedded projects are software projects that must be developed within a set of tight hardware, software and operational constraints.

This version of COCOMO is the most simple and only captures a few relations that influence the effort and cost in software projects [8].

B. Intermediate COCOMO

The original COCOMO model had some disadvantages which were improved in the COCOMO II version (2000). The latter's objectives are (1) Provide accurate cost and schedule estimated for both current and likely future projects, (2) Enable organizations to easily recalibrate, tailor and extend COCOMO II to better fit their unique situations, (3) Provide careful, easy-to-understand definitions of the model's inputs, outputs and assumptions, and (4) Provide a constructive, normative, evolving model. In addition to the model's objectives, the model was adapted to account for (1) non-sequential and rapid development process models, (2) reuse driven approaches using COTS packages, reengineering, application compositions and application generation capabilities, (3) object-oriented approaches supported by distributed middleware and (4) software process maturity effects and process-driven quality estimation [9]. A major improvement in this model is that new cost factors have been added and that the model has been calibrated on a larger sample data and with more recent project data. Detailed information can be found in Software Cost Estimation with COCOMO II by Boehm et. al. [10].

C. Walston-Felix Model

The basic equation of Walston and Felix model (1977) is:

$$Effort = 5.2\ KLOC^{0.91} \quad (2)$$

For this equation, they gathered 60 sample projects from IBM. These projects differed widely in size and programming languages used for the development of the software. It then comes as no surprise that the model, when applied to a subset of these 60 projects, produces unsatisfactory results. To explain these wide-ranging results, Walston and Felix identified 29 variables that clearly affect productivity. For each variable, three levels were determined: high, average and low. For a number of projects Walston and Felix determined the level of each of these 29 variables, together with the productivity acquired, which is in terms of lines of code per man-month, in those projects [3]. The basic equation was supplemented with a productivity index that reflected 29 factors that can affect productivity. The factors are tied to a very specific type of development, including two platforms: an operational computer and a development computer. The model reflects the particular development

29th November 2012. Vol.7 No.1

ISSN 2304-0777 www.jitbm.com

style of the IBM Federal Systems organization that provided the data [2].

D. Bailey-Basili Model

Bailey and Basili (1981) suggested a modeling technique, called a meta-model, for building an estimation equation that reflects an organization's characteristics. They demonstrated their technique using a database of 18 scientific projects written in Fortran at NASA's Goddard Space Flight Center. They minimized the standard error estimate and produced this equation [2]:

$$\text{Effort} = 5.5 + 0.73\, S^{1.16} \quad (3)$$

Then they adjusted the initial estimate based on the ration of errors. If *R* is the ration between the actual effort, *E*, and the predicted effort, *E'*, then the effort adjustment is defined as:

(4)

$$ER_{adj} = \begin{cases} R - 1, & \text{if } R \geq 1 \\ 1 - 1/R, & \text{if } R < 1 \end{cases}$$

Then the initial effort *E* was adjusted this way:

(5)

$$E_{adj} = \begin{cases} (1 + ER_{adj})E, & \text{if } R \geq 1 \\ E/(1 + ER_{adj}), & \text{if } R < 1 \end{cases}$$

Finally, they accounted for other factors that affect effort. Their model describes a procedure, based on a multilinear least-square regression, for using these effort factor scores to further modify the effort estimates [2].

III. METHODOLOGY

A. Data Source and Data Collection

The source for the project data for this study is a global application development outsourcing company specializing in business and finance-related software applications. For data retrieval, software metrics for XYZ's completed projects, such as size (in LOC), start and end dates, effort (in person-months) and productivity (in LOC/person-months) were collected and filtered. These data were available as spreadsheet files obtained from the Project Manager. Personnel directly involved in project estimation were interviewed and asked for background information about the projects they developed, how they estimated past projects, what methods they used in estimating project, what factors they considered during estimation and if their estimates were accurate. Questionnaires were prepared and project manager and project leaders were interviewed. To measure the physical size of a project, the source lines of code were taken into account as well as the actual effort and duration for a particular project.

B. Empirical Validation of the Software Cost Models

The metrics data (SLOC) was used as input for the basic COCOMO, intermediate COCOMO, Walston-Felix, SLIM and Bailey-Basili models. The degree to which the model's estimated effort ($MM_{estimated}$) matches the actual effort (MM_{actual}) undergoes evaluation. For the basic COCOMO, each project was classified by their development mode – whether they are organic, semi-detached or embedded. For the intermediate COCOMO, each cost driver in every category was given a rating to adjust the baseline development effort estimation. For Walston-Felix and Bailey-Basili models, input value are the SLOC and output was the estimated effort in person-months.

To eliminate the problem caused by project size and better reflect the impact of any errors, a percentage error test (as recommended by Boehm and others) was performed, as follows:

(6)

$$PercentageError = \frac{MM_{estimated} - MM_{actual}}{MM_{actual}}$$

To determine the average performance over the entire set of projects and to measure the accuracy of a project estimate, the mean of relative error (as suggested by Conte et. al. 1986) or MRE was computed as follows:

(7)

$$MRE = \left| \frac{MM_{estimated} - MM_{actual}}{MM_{actual}} \right|$$

To check whether the estimates from every model correlate with the actual results, linear regression was done (as proposed by Albrecht and others). To develop a regression model, the actual man-months was used as the dependent variable and the man-months estimated by each model is the independent variable. This method can

29th November 2012. Vol.7 No.1

ISSN 2304-0777 www.jitbm.com

show whether a model's estimates correlates well with experience even when the MRE test does not. An error percentage test of 0 on the MRE test would be a "perfect" score, whereas an R^2 of 1.00 on Albrecht's regression test would be a "perfect" score.

Another measure to examine the cumulative frequency of MRE for a specific error level is called PRED (r) or Prediction at level r. If k is the number of projects in a set of n projects whose MRE ≤ r, then PRED (r) = k/n. According to Conte, Dunsmore and Shen (1986), a value of PRED (25%) ≥ 75% and MRE ≤ 25% are considered desirable for effort models.

C. Designing the Effort Estimation Model

Estimation models suited to XYZ was developed for this study. A simple linear regression model was done, in which the effort is the variable to be predicted, and the source lines of code will be the main independent factor. Another model taken into consideration is the different project types (whether it is a web-based application, a mobile application, etc.) and other attributes affecting the project. This was done by using linear regression with categorical variables. Another model will be created using logarithmic transformation, in which the logs of y (effort) and x (LOC) was used to derive the model, resulting in the equation form of $E = a\ LOC^{b}$. Cost factors affecting productivity were also considered and incorporated into the model.

Following Boehm's definition of SLOC, it is defined such that (1) Only source lines that are delivered as part of the product are included while test drivers and other support system are excluded, (2) Source lines are created by the project staff while code created by applications generators is excluded, (3) One SLOC is one logical line of code, (4) Declarations are counted as SLOC and (5) Comments are not counted as SLOC.

D. Empirical Validation of the Proposed Cost Models

Aside from the validations mentioned in Section 3.2., an analysis of variance (ANOVA) was done to measure the performance and significance of the model. A scatter plot was generated to see if a linear relationship between effort and lines of code exists, and to see if a linear regression, regression with categorical values or a logarithmic transformation is plausible. A residual plot was developed to check the normality of the residuals.

IV. RESULTS AND DISCUSSION

A. Data Description

There were a total of 63 projects gathered from the year 2006 to 2009. These projects range from 3 KLOC to 1172.14 KLOC. Actual effort varies from 0.92 PM to 234 PM. The project type specifies the type of project, whether it is a web application, a mobile application, or a stand-alone/desktop application among others. The contract specifies the agreement between the company and the client with regards to the schedule and duration of the project. If a contract is project-based, the company makes an estimate of the entire project and builds a schedule from the estimates made. On the other hand, if a contract is time and materials, the client will decide on the schedule and the number of people who will work on that particular project. The KLOC values follow Boehm's definition; in which comments are not to be included counting the lines of code. Since the client made the decision regarding the duration of the project for time and materials contract project, no estimations were done for time and materials contract projects. Hence, they were filtered out during the model analysis and estimation model development.

B. COCOMO Results

Both COCOMO models did poorly based on the MRE percentage error test. The mean error percentage for the Basic COCOMO is 9,217.62 percent and for the Intermediate COCOMO, the error percentage is 5,818.75 percent. Since both models have MRE values > 25%, they are not considered as a desirable model to use for XYZ. The PRED (25%) for the Basic and Intermediate model is only 1.82 %. For the basic model, the lowest single error percentage is 18.28 percent and the highest is 268,588.34 percent. For the intermediate model, the lowest single error percentage is 20.16 percent and the highest is 167,512 percent. There are very few cases in the basic COCOMO values that underestimated the actual values, but overall, effort estimates are biased and are overestimated.

The Intermediate model has a lower error percentage with a value of 5,818.75 compared to the Basic model with value of 9,217.62. This implies that the cost drivers of the Intermediate model are not providing any additional explanations of this phenomenon and they do not necessarily contribute to estimating project effort. It is possible that this model accurately reflected the project conditions, but is calibrated too high for the environment of XYZ.

29th November 2012. Vol.7 No.1

ISSN 2304-0777 www.jitbm.com

Albrecht's regression test was also done, in which linear regression was made, using the COCOMO estimates as the independent variable and the actual effort as the dependent variable. The coefficient of determination R^2 – which provides a measure of how well future outcomes are likely to be predicted by the model – was also computed. The results were as follows:

COCOMO I:
Actual PM = 25.59 + 0.003 (COCOMO I) with r^2 = 4%

COCOMO II:
Actual PM = 25.56 + 0.003 (COCOMO II) with r^2 = 9%

COCOMO II, the advanced version of the two, seems to be adding a bit of information as its r^2 slightly increased, but since both r^2 values are very low, they are not the right fit. Both models have very low accuracy in estimating effort for the XYZ environment.

C. Walston-Felix Results

The error percentage value shows that this model did slightly better than COCOMO I and COCOMO II – with an error percentage of 4666.389 percent – lower than the Basic and Intermediate COCOMO. For this model, the lowest single error percentage is 18.48 percent and the highest is 105861.17 percent. The model's PRED (25%) is 1.81%, same as with the Basic and Intermediate COCOMO values.

Although the model performed better that COCOMO I and COCOMO II, this does not mean that the Walston-Felix model can be used with utmost confidence, since its mean relative error is higher than 25 percent. One possible explanation for this is that the model is based on IBM sample projects with different languages and application types, which may produce unsatisfactory results. An ideal MRE rating would be a value of zero, which means there is no error at all, or near zero value, which means there is a small error.

Albrecht's linear regression test was also done in which linear regression was made using the Walston-Felix estimates as the independent variable and the actual effort as the dependent variable. The results were as follows:

Actual PM = 25.27 + 0.004 (Walston-Felix) with r^2 = 4%

This model did not do as well as the COCOMO II in this instance. The r squared value is still very small, which means it is not as good a fit as with the previous models evaluated.

D. Bailey-Basili Results

The lowest single error for this sample is 9.18 percent and the highest error percentage is 2656.07 percent. The mean error percentage for this model is 2895.76 percent, which is lower than the Walston-Felix model and the Basic and Intermediate COCOMO but still greater than 25 percent; this implies that the model cannot be used for XYZ. The PRED (25%) for this model 5.45 percent, an improvement from the previous models but still below the acceptable value which is 75 percent.

The regression analysis result from the XYZ data, having the Bailey-Basili model's values, is specified below:

Actual PM = 25.74 + 0.004 (Bailey-Basili) with r^2 = 3%

This model has the lowest mean error percentage, but also has the smallest R^2 value. Therefore, this model, along with the previous models evaluated, does not seem to be adding additional information.

E. Model Performance Summary

Table 2 summarizes the performance of models being evaluated. All four models failed in the MRE test, PRED (25%) and Albrecht's r^2 test. Lowest MRE is the Bailey – Basili Model having a value of 2895.76. Highest value of PRED (25%) is the Bailey – Basili Model with 5.45 percent. COCOMO II has the highest r2 value of 9 percent. Table 2 shows that the model that has the best performance is Bailey-Basili and the worse performance came from COCOMO I.

Table 2. Summary of Software Cost Model Performances

	MRE	PRED (25%)	Albrecht's r^2
COCOMO I	9217.62	1.81	0.04
COCOMO II	5818.75	1.81	0.09
Walston–Felix	4666.38	1.81	0.04
Bailey–Basili	2895.76	5.45	0.03

F. Model Development

1. First Model

This model was formulated using linear regression with categorical variables; only this time there is only one

29th November 2012. Vol.7 No.1

ISSN 2304-0777 www.jitbm.com

dummy variable. The model still takes into consideration the project type, only the classification is either web app or not a web app. The equation is as follows:

(10)

$$E = 41.469 + 0.017\ KLOC - 29.173X_0$$

where X_0 = web application (1 if applicable, 0 otherwise)

For this model, the value of R is 0.481, the R squared is 0.232, the adjusted R squared is 0.201, the standard error of the estimate is 24.393 and the Durbin-Watson is 1.776.

An analysis of variance was also done to measure the performance and significance of the model. About 23.17 percent is accounted for by the model, and the significance (ANOVA) is 0.001, which implies that this particular model is statistically significant.

2. Incorporating Cost Factors

For this model, cost factors such as programmer capability (PCAP) and use of software tools (TOOLS) are included. These cost factors are based on Boehm's COCOMO II Software Cost Drivers.

Programmer Capability (PCAP) represents the capability of the programmers who will be working on the software product. Ratings are expressed in terms of percentiles with respect to the overall population of programmers. Major factors that should be considered in the rating are ability, efficiency, thoroughness and the ability to communicate and cooperate. This should be based on the capability of the programmers as a team rather than as individuals. This factor can be rated as very low (15th percentile), low (35th percentile), nominal (55th percentile), very high (90th percentile) or extra high (no rating – defaults to very high).

Use of software tools (TOOLS) represents the degree in which software tools are used in developing the software products. This factor can be rated as very low (basic tools, e.g. assembler, linker, monitor, debug aids), low (beginning use of more productive tools, e.g. High-order language compiler, macro assembler, source editor, basic library aids, database aids), nominal (some use tools such as real-time operating systems, database management system, interactive debuggers, interactive source editor), high (general use of tools such as virtual operating systems, database design aids, program design language, performance measurement and analysis aids, and program flow and test analyzer), very high (general use of advanced tools such as full programming support library with configuration management aids, integrated documentation system, project control system, extended design tools and automated verification system) and extra high (no rating – defaults to very high).

The cost models are then incorporated into the first model, with which the following equation is formed:

(12)

$$E = 41.469 + 0.017KLOC - 29.173X_0 * PCAP * TOOLS$$

where X_0 = web application (1 if applicable, 0 otherwise)

Programmer Capability (PCAP) and Use of software tools (TOOLS) rated as very low, low, nominal, high, very high or extra high

Table 4 below shows the corresponding numerical values per rating.

Table 4. Effort Multipliers for Software Cost Factors

Cost Drivers	VL	L	N	H	VH	EH
PCAP	1.42	1.17	1.00	0.86	0.70	0.70
TOOLS	1.24	1.10	1.00	0.91	0.83	0.83

Table 5 shows the summary of all model performances. Out of the 6 models produced, the 4th model had the lowest MRE with value of 195.11 percent and highest PRED (25%) with value of 10.91 percent. When the model was incorporated with cost factors, the level of accuracy decreased, having an MRE value of 239.99 percent and PRED (25%) at 10.90 percent, which implies that adding cost factors does not seem to improve accuracy.

Table 5. Summary of all model performances

	Mean Value	MRE	Lowest MRE	Highest MRE	PRED (25%)
COCOMO I	704.64	9217.62	14.29	8210.65	1.81
COCOMO II	505.09	5818.57	11.35	8209.57	1.81
Walston-Felix	681.91	4699.39	18.49	3226.19	1.81

29th November 2012. Vol.7 No.1

ISSN 2304-0777 www.jitbm.com

Bailey-Basili	552.05	2895.77	9.18	2656.07	5.45
1st model	21.68	195.11	8.41	1143.20	10.91
1st model with cost factors	19.21	239.99	2.11	1635.74	9.09

V. SUMMARY AND CONCLUSION

All four models, namely COCOMO I, COCOMO II, Walston-Felix and Bailey-Basili did not do well on the MRE and PRED (25%) test; none of them were particularly effective with our project data. A possible explanation for this is that these models were derived by studying large number of completed software projects from various organizations and applications. Since these models are developed under different environments, it comes as no surprise that they do not work well for XYZ. Since the COCOMO I, Walston-Felix and Bailey-Basili models do not consider other cost factors, they may only be used as a rough estimate. COCOMO II, on the other hand, having accounted for some cost drivers, ironically had a bad performance in estimating effort. A possible explanation for this is that COCOMO II used project data varying in application types. Using real project data and developing estimation models from it seems to improve accuracy performance; however, much work still needs to be done in order to improve and gain more accurate estimates. Adding cost factors from Boehm's defined cost drivers in COCOMO II, particularly programming capability and use of software tools does not seem to improve estimation accuracy. At the confidence level of 25%, there is a guarantee of about 11% accuracy. The model can be used for deriving estimates but do so with caution. The model cannot guarantee good estimates for non-web application projects, since 75% of the project data are web application types. XYZ project estimators can use the estimates derived from this model to coincide with their WBS, FP and expert judgment estimates. However, it is advised not to rely solely on the estimates generated by the model, since it does not account for other factors such as the complexity of the software to be developed. Nonetheless, it is important to point out that the model developed has more accurate and realistic results compared to the results of the other existing software cost models. Unlike expert judgment of WBS, an advantage of using this model is that it is quantifiable and can be based on ex-post data.

VI. RECOMMENDATION

It would be ideal to consider other size metrics aside from lines of code. Since majority of the project being developed is web applications, it would be best to focus on creating a model that can estimate the effort needed for web application types of project. An ideal metrics aside from LOC would be page count, media count, program count, and reused modules, among others. Since project data gathered is very limited (the only available information per project is the KLOC, actual effort, and project type), the model created is also very limited. Data collected from the company seems incomplete, since they have a skill map for every employee and they also have a list of technologies and platform used, but there was no associated developer(s) and technology used per project. Therefore, the model is mainly limited to the available data. Identifying and quantifying other factors that affect productivity based on the environment of XYZ can further refine the model and produce more accurate estimates.

ACKNOWLEDGEMENT

The author would like to thank DOST-ASTHRDP for the financial assistance extended throughout the duration of this study. Additional thanks to Ralf Mosqueda, Sheila Echiverri and Pauline Wade for permission to retrieve data needed for the study.

REFERENCES

[1] Fairley, R. Software Engineering Concepts International Edition. McGraw Hill Book Co. 1985.

[2] Pfleeger, S. L. Software Engineering Theory and Practice International Edition. Prentice Hall, Inc. 1999.

[3] van Vliet, Hans. Software Engineering Principles and Practice Second Edition. John Wiley & Sons, Ltd. 2000.

[4] CSSE Website http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html

[5] Overview of COCOMO http://www.softstarsystems.com/overview.htm

[6] Boehm, B. Software Engineering Economics. Englewood Cliffs, NJ: Prentice-Hall, 1981.

29th November 2012. Vol.7 No.1

ISSN 2304-0777 www.jitbm.com

[7] M. Giombetti, An Analysis of Software Cost Estimation Methods with Regards to Quality Requirements, Technische Universitat Munchen, 2010.

[8] Software Measurement, Cost Estimation, SLIM, COCOMO http://yunus.hacettepe.edu.tr/~sencer/cocomo.html

[9] M. Giombetti, Cost/Benefit-Aspects of Software Quality Assurance - Selected Topics in Software Quality, Report: TUM-I0824: Institut für Informatik - Technische Universität München, 2008.

[10] Boehm B., Abts, C., Brown, A.W., Chulani, et. al., Software Cost Estimation with Cocomo II, Prentice Hall PTR, 2000.