Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 59A (2013) 15811-15815

ISSN: 2229-712X

Corean: A proposed Model for Predicting Effort Estimation having Reuse

Jyoti Mahajan and Simmi Dutta

Department of Computer Engineering, Govt. College of Engg. & Tevhnology, Jammu.

ARTICLE INFO

Article history:

Received: 20 December 2012;

Received in revised form:

10 June 2013;

Accepted: 14 June 2013;

Keywords

Effort Estimation,
Software Reuse,
COCOMO II,
Artificial Neural Network,
Simulated Annealing.

ABSTRACT

The estimation accuracy has been focused in various formal estimation models in recent research initiatives. The formal estimation models were developed to measure lines of code and function points in the software projects but most of them failed to improve accuracy in estimation. The concept of reusability in software development in estimating effort using artificial neural network is focused in this paper. Incorporation of reusability metrics in COCOMO II may yield better results. In COCOMO II it is very difficult to find the values of size parameters. A new model called COREAN has been proposed in this paper for better effort estimation accuracy and reliability. The proposed model has focused on two components of COCOMO II. First, instead of using RUSE cost driver, three new reuse cost drivers are introduced. Second, In order to reduce the project cost, three cost drivers such as PEXE, AEXE, LTEX are combined into single cost driver Personnel Experience (PLEX). Finally, this proposed model accuracy is more improved with the help of Enhanced RPROP algorithm and simulated annealing optimization technique.

1. Introduction

Most of the software development projects were failed due to effort overrun and exceeding its original estimates as per the survey conducted in various research publications [1]. Effort overruns usually lead to cost overruns and missed project deadline. In software engineering estimating software development effort is one of the most critical and complex task. Over the last three decades, a growing trend has been observed in using variety of software effort estimation models in diversified software development processes. Along with this tremendous growth, it is also realized that the essentiality of all these models in estimating the software development costs and preparing the schedules more quickly and easily in the anticipated environments. Although a great amount of research time and money have been invested to improve the accuracy of the various estimation models. Due to the inherent uncertainty in software development projects such as complex and dynamic interaction factors, change of requirements, intrinsic software complexity, pressure on standardization and lack of software data, it is unrealistic to expect very accurate effort estimation of software development processes[2].

Reusability has benefits such as reduced effort, improved productivity, decreased time-to-market and decreased cost in software development. This research work addresses the significance of reusability in effort estimation and formulates new metrics for reusability to determine the reliable and accurate effort estimates. Selecting an appropriate model for a specific project is an issue in project management[3]. The appropriate model which provides minimum relative error has to be considered as the best fit for effort estimation.

2 Related Works

2.1 Extensions of COCOMO II

The COCOMO II [4][5] project was started to meet the future requirements of the next generation of software development process. The new COCOMO II model has incorporated features that are realistic and accurate in COCOMO 81 and Ada COCOMO models. COCOMO II has proposed three submodels based on development stages of the project. The Application Composition model is the first submodel used to estimate effort and schedule on projects that use rapid application development tools. Early design model is used to get approximate estimate in the preliminary stages of the project. Post architectural model is mainly used to estimate effort when the high level design is completed. COCOMO II defined the reuse model which adjusts the code reuse by modifying the size of the module or project. This model considers reuse with function points and source lines of code the same in either the early design model or the post-architecture model. A size estimate equivalent to the number of lines of new source code is computed and then adjusts the size estimate for new code. This model has not clearly specified complete system to evaluate the “actual” equivalent SLOC. It is difficult to calibrate the model and difficult to determine the parameters Design Modified (DM), Code Modified (CM), reuse software (IM) and Adapted SLOC.

Estimating development effort using reuse proposed by Balda and Gustafson [6]. This model adapted the simple COCOMO model by distinguishing newly developed code that is specific to the project, newly developed code that is made for reuse and code that is modified for reuse. This model uses the four variables to represent these types of code.

COCOMO II Constructive Staged Schedule & Effort Model (COSSEMO) [7] specifies the percentages of effort and schedule to be applied to the different stages of project: Inception, Elaboration and Construction. The predicted effort and schedule from a COCOMO II correspond to the sum of effort and schedule of inception, Elaboration and Construction stages. Thus, the sum of the effort or schedule for three stages can actually total more than 100% of the COCOMO II effort and schedule.

Tele:
E-mail addresses: jyoti_1972@sify.com

Constructive RAD Schedule Estimation Model (CORADMO) [8] model has five drivers. Each driver has both rating levels, which are selected by a user based on the characteristics of the software project, its development organization, and its milieu. There are numeric schedule and effort multiplier values per stage for each rating level. The impact of re-use of 3GL production code is handled directly in the COCOMO II model via the re-use sub-model and its effect on size. This CORADMO driver reflects the impact of re-use of code and/or the use of very high level languages, especially during the Inception and Elaboration stages. Higher rating levels reflect the potential schedule compression impacts in Inception and Elaboration stages due to faster prototyping, option exploration. Clearly this impact will be dependent on the level of capability and experience in doing this, such as Rapid Prototyping experience. The values of the multipliers corresponding to the rating levels are the same for both effort and schedule; this implies that the staff level is held constant.

Constructive Quality Model (COQUALMO) [9] is an extension of the existing COCOMO II model to specify the quality. This model is based on the software defect introduction and removal model described by Barry Boehm. The defects conceptually flow into a holding tank through various defect source pipes. These defect source pipes are modeled in COQUALMO as the "Software Defect Introduction Model". The Defect Introduction and Defect Removal Sub-Models described above can be integrated to the existing COCOMO II cost, effort and schedule estimation model.

Constructive COTS integration cost model (COCOTS) [10] where COTS in turn is short for commercial-off-the-shelf, and refers to those pre-built, commercially available software components that are becoming ever more important in the creation of new software systems. This model was developed as an extension of the COCOMO II cost model for reusable components based software development effort estimation. COCOTS attempts to predict the lifecycle costs of using COTS components by capturing the more significant COTS risks in its modeling parameters.

The primary approach modeled by COCOMO is the use of system components that are developed from scratch or new code. But COCOMO II also allows you to model the reusability in which system components are built out of pre-existing source code. Even most the projects are not building the reuse component from scratch but reusable component's source code can be modified to suit your needs. COCOMO II currently does not model the case in which project has access to a pre-existing component's source code.

2.2 ANN based Effort Estimation

Literature reveals that many software engineering researchers have proposed ANN based approach to estimate software development effort [9,10, 11, 12, 13]. The back propagation trained multilayered feed forward networks is generally used in most of the research work to predict the software effort estimation. The use of ANN with a back propagation learning algorithm for effort estimation has explored [11,14,15] and found the effectiveness of the neural network technique in effort estimation. Some preliminary investigation in the use of neural network in estimating software cost and produced very accurate results[11], but the major set back in their work was due to the availability of dataset and the accuracy of the result depends on the size of the training set.

3. Proposed Model - COREAN

The proposed model is estimating more accuracy and reliable software effort with the help of software reusability concept. Comparing with COREAN, Software reusability in COCOMO II is not provided an accuracy result. Instead of RUSE cost driver, three new reuse cost drivers is introduced such as Reuse Veryhigh Level Language (RVLL), Required Integrator for Product Reuse (RIPR), Reuse Application Generator (RAPG) is yielding best result for reusability in software effort estimation. The effort estimation formula of **COREAN** is,

$$PM = 2.94*(SIZE)^{B} * \prod_{i=1}^{17} EM_i \quad \text{-----1}$$

Where,

$$B = 0.91 + 0.01*\sum_{j=1}^{5} SF_j \quad \text{----------2}$$

The COREAN model Scale Factors are same as the COCOMO II [7][8] model scale factors such as PREC, FLEX, RESL, TEAM, PMAT.

$$SIZE = \left(1+\frac{REVL}{100}\right)*(\text{New KSLOC} + \text{Equivalent KSLOC}) \quad \text{----------3}$$

$$\text{Equivalent KSLOC} = \text{Adapted KSLOC}*\left(1-\frac{AT}{100}\right)*AAM \quad \text{----------4}$$

$$\text{Where } AAM = \begin{cases} \dfrac{AA + AAF*(1+[0.02*SU*UNFM])}{100} & \text{, For AAF <=50} \\ \dfrac{AA + AAF + (SU*UNFM)}{100} & \text{, For AAF > 50} \end{cases}$$

$$AAF = (0.4)*DM + (0.3)*CM + (0.3)*IM \quad \text{-----5}$$

COREAN Cost Drivers:

- Product reliability and complexity - RELY, DATA, CPLX, DOCU
- Required reuse - RVLL, RIPR, RAPG
- Platform difficulty - TIME, STOR, PVOL
- Personnel capability - ACAP, PCAP, PCON
- Personnel experience – PLEX
- Facilities - TOOL, SITE
- Required Development Schedule - SCED

3.1 New Metrics Introduction

Three cost drivers such as PEXE, AEXE, LTEX are combined into single cost driver Personnel Experience (PLEX) for reducing the software project cost.

Instead of RUSE metric in COCOMO II, three new reuse metrics are introduced,

1)RVLL(Reuse Very high Level Language)
2)RIPR(Required Integrator for Product Reuse)
3)RAPG(Reuse Application Generator)

3.2 New Metrics Definition and Validation Methodologies

The Goal/Question/ Metric (GQM) paradigm provides a template and guidelines to define metric goals and refine them into concrete and realistic questions, which is subsequently lead to the definition of measures. Software engineering process requires feedback and evaluation mechanism to define and validate metrics. GQM is usable as a practical guideline to design and reuse technically sound and useful measures. It provides templates for defining goal and generate questions to

define new metrics in software engineering process[16][17].The main focus is to construct cost drivers for predictive models that establish a reliable effort estimation. Goals are defined in an operational way by refining them into a set of quantifiable questions that are used to extract appropriate information. The new cost drivers are defined under GQM methodology.

These new cost drivers are properly validated with the help of Theoretical (Internal) validation and Empirical (External) validation [18][19]. The important of theoretical validation is to measure and asses the metric intensions using DISTANCE framework[20] and the empirical validation by gathering the information about the metrics using survey method. To validate the EAF of proposed model, company dataset containing 20 project has been used. By adjusting the value of cost drivers, this will yield better result than past projects.

3.3 COREAN with ANN Model Implementation:

To implement ANN model, COREAN effort estimation Equation 1 should be transform from non linear model to linear model by applying natural logarithm on both sides. ANN is implemented with Enhanced RPROP[21].

$$In(PM) = In(A) + 0.91 * In(SIZE) + SF_1 * 0.01 * In(SIZE) + \ldots\ldots\ldots + SF_5 * 0.01 * In(SIZE) + In(EM_1) + In(EM_2) + \ldots\ldots\ldots + In(EM_{17}) \quad \text{---------- 6}$$

[Linear Equation]

$$OP_{est} = WT_0 + WT_1 * IP_1 + WT_2 * IP_2 + \ldots + WT_6 * IP_6 + WT_7 * IP_7 + \ldots + WT_{23} * IP_{23} \quad \text{---------- 7}$$

[ANN Based Model For Effort Estimation]

Actual observed effort is compared with this estimated effort. The differences between these values are the error in the effort. It should be minimized.

3.4. SA Optimization

In the proposed model COREAN, Simulated Annealing Algorithm[22] is used to estimate the optimum solution of the software project effort. The given solution method is helped to get optimal values of effort:

$$\text{Minimize} \sum_{i=1}^{n} (Effort_M - Effort_C)^2 \quad \text{----------8}$$

Where, $Effort_M$ = Measured Value of Effort, $Effort_C$ = Computed Value of Effort according to the model used.

Simulated Annealing Algorithm Procedure:

1. Initialization: parameters of annealing schedule.
2. Select an iteration mechanism: a simple prescription to generate a transition from current state to another state by a small perturbation.
3. Evaluate the new state, compute the value of ΔE = (value of current state - value of new state).
4. If the new state is better, make it current state, otherwise probabilistically accept or reject it with a determined probability function
5. if condition is true continue Step 2 otherwise terminated.

4. Results

Out of the 20 project dataset, to forecast an effort of the proposed model. The estimated effort is comparing with existing COCOMO II and Actual effort of the project. This results are shown as Table – 1 and comparison graph also provided as below:

Table - 1 : Comparison of Effort Estimation With SA Optimization

Project ID	Actual Effort	Estimating Effort (PM) using	
		COCOMOII	COREAN with SA Optimization
1	205	117.6	192
2	211	117.6	173
3	40	31.2	32.6
4	24	36	25.15
5	43	25.2	44.48
6	15	8.4	4.87
7	9	10.8	13.9
8	36	25.2	38.12
9	277	352.8	254
10	95	72	104.1
11	67	72	101.87
12	39	24	22.7
13	255	360	259
14	77	36	79.2
15	288	215	287.3
16	345	360	315
17	398	360	407
18	299	324	303.8
19	102	60	89
20	76	48	61

Figure - 1 : Comparison of Effort Estimation

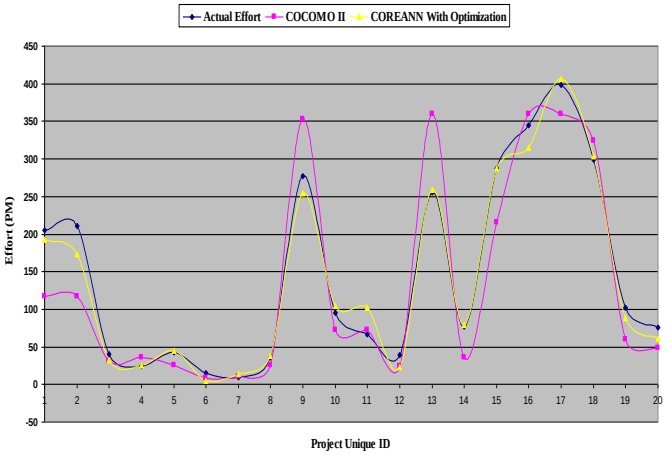
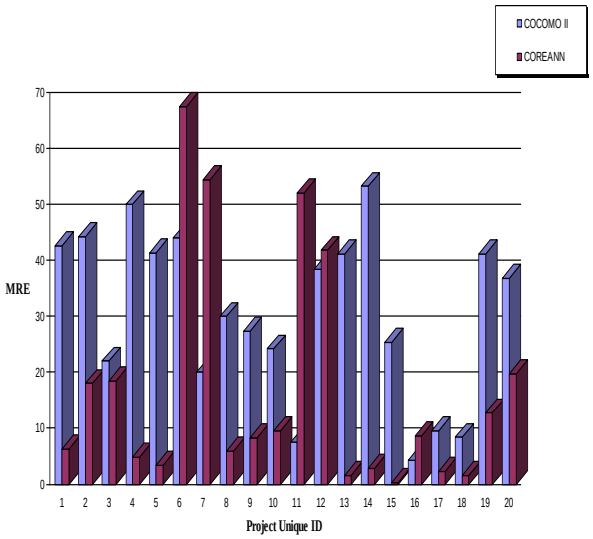


Table – 2 shows that the result for MRE comparison of the proposed model with existing COCOMO II.

Table 2 - Comparison of Effort Estimation Results In MRE

Project ID	MRE using	
	COCOMO II	COREAN
1	42.63	6.34
2	44.27	18.009
3	22	18.5
4	50	4.792
5	41.39	3.44
6	44	67.53
7	20	54.44
8	30	5.89
9	27.36	8.303
10	24.21	9.58
11	7.46	52.05
12	38.46	41.795
13	41.18	1.769
14	53.25	2.857
15	25.35	0.243
16	4.35	8.696
17	9.58	2.26
18	8.361	1.605
19	41.18	12.75
20	36.84	19.74
$MMRE_{COCOMOII}$ = 30.592 $PRED(25)_{COCOMOII}$ = 35.00 $MMRE_{COREAN}$ = 17.019 $PRED(25)_{COREAN}$ = 80.00		

Figure - 2 : MRE Comparison



In Table – 2, MMRE value of COREAN is 17.019 and COCOMO II is 30.592. $PRED_{(25)}$ value of COREAN is 80.00 and COCOMO II is 35.00. By the above result, observed value for MMRE of COREAN is less than MMRE of COCOMO II and $PRED_{(25)}$ of COREAN is greater than $PRED_{(25)}$ of COCOMO II.

5. Conclusion and Future work

In software engineering, it is extremely difficult to select appropriate model for estimation effort estimation due to the availability of number of models. Software reuse has become a major factor in development. Hence, effort estimation for reuse must accurate for the successful project execution. This paper primarily concentrated on the computation of accurate effort with software reusability as the main focus. While comparing performance results of COREAN and COCOMO II, it clearly shows that the proposed COREAN works better than COCOMO II. That is, the COREAN model is estimated lower MRE & MMRE and higher PRED(25) than the COCOMO II model. So the prediction accuracy of COREAN is high based on the performance evaluation. In future work, the effort estimated by expert judgment method has to be considered to optimize the final effort estimation. Initial value of the optimization is the effort estimated by expert judgment.

References

[1] K. Molokken-Ostvold and M. Jorgensen, “A Review of Surveys on Software Effort Estimation,” Proc. 2003 ACM-IEEE Intternational Symposium on Empirical Software Eng, pp. 220-230, 2003.

[2].Saleem Basha and Dhavachelvan P, “Analysis of Empirical Software Effort Estimation Models”, International Journal of Computer Science and Information Security, Vol. 7, No. 3, 2010

[3] Chao-Jung Hsu, Nancy Urbina Rodas, Chin-Yu Huang and Kuan-Li Peng “A Study of Improving the Accuracy of Software Effort Estimation Using Linearly Weighted Combinations”, 34th Annual IEEE Computer Software & Application Conference Workshops, 2010

[4] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and 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, Science Publishers, vol. 1, pp. 45-60, 1995.

[5] Boehm, B. COCOMO II Model Definition Manual. Center for Software Engineering, University of Southern California. 1997.

[6] Balda, D,. M. and D. A. Gustafson, "Cost Estimation Models for the Reuse and Prototype Software Development Life-Cycles", ACM Sigsoft Software Engineering Notes, 15 (3), pp. 4250, 1990.

[7]A.Windsor Brown url: http://csse.usc.edu/publications/TECHRPTS/KBSA_Tech_Report/app5.pdf, 1999

[8] Barry Boehm, A.Winsor Brown, http://csse.usc.edu/publications/TECHRPTS/KBSA_Tech_Report/volumeI.pdf, 1999

[9] Sunita Chulani, Barry Boehm "Modeling Software Defect Introduction Removal: COQUALMO (COnstructive QUALity MOdel)", Technical Report USC-CSE-99-510, 1998

[10] C.Abst, B.Boehm, E.Clark. “COCOTS: A COTS Software Integration Lifecycle Cost Model - Model Overview and Preliminary Data Collection Findings", Technical report USC-CSE-2000-501, USC Center for Software Engineering, 2000.

[11] Heiat A, “Comparison of artificial neural network and regression models for estimating software development effort,” Journal of Information and Software Technology, Volume 44, Issue 15, Pages 911-922, 2002.

[12] Wittig, G., Finnie, G., “Estimating software development effort with connectionist models”, Information and Software Technology, 39 (7), 469–476, 1997

[13] Karunanithi, N., D. Whitely, Y. K. Malaiya, “Using neural networks in reliability prediction”, IEEE Software, pp. 53-59, 1992.

[14]Tadayon, N., “Neural network approach for software cost estimation,” International Conference on Information Technology: Coding and Computing (ITCC 2005), Volume: 2, on page(s): 815- 818, 2005.

[15]Dawson, C.W., “A neural network approach to software projects effort estimation,” Transaction: Information and Communication Technologies, Volume 16, pages 9, 1996.

[16]. Lionel C. Briand, Sandro Morasca, and Victor R. Basili, “An Operational Process for Goal-Driven Definition of Measures”, IEEE Transactions On Software Engineering, Vol. 28, No. 12, 2002.

[17] V. Basili, "Software Modeling and Measurement: The Goal/Question/Metric Paradigm" University of Maryland, Department of Computer Science, Tech. Rep. CS-TR-2956, 1992.

[18] L. Briand, K. El Emam, S. Morasca, “Theoretical and Empirical Validation of Software Product Measures”, Technical Report ISERN-95-03, Fraunhofer Institute for Experimental Software Engineering, Germany, 1995.

[19] Murat Ayyıldız, Oya Kalıpsız, and Sırma Yavuz, “A Metric-Set and Model Suggestion for Better Software Project Cost Estimation”, World Academy of Science, Engineering and Technology, 2006.

[20] G. Poels, G. Dedene, DISTANCE: A Framework for Software Measure Construction, Reserch Report 9937, Dep. of Applied Economics, Katholieke Universiteit Leuven, 1999.

[21] Jyoti Mahajan, Devanand, “Reusability in Effort Estimation model based on Artificial Neural Network for Predicting Effort in Software Development”, Research Cell : An International Journal of Engineering Sciences, vol. 4, 2011.

[22] Mitat Uysal, "Estimation of the Effort Component of the Software Projects Using Simulated Annealing Algorithm", World Academy of Science, Engineering and Technology ,2008

Dr. Jyoti Mahajan obtained his B.E. Degree in Computer Science & Engineering from Bangalore University, M.S. (Software Systems) from BITS Pilani and Ph.D. in Computer Science from University of Jammu. He is working as Lecturer in the Department of Computer Engineering at GCET, Jammu, India. He has about 12 years teaching experience. His Research interests include Data Structure, Software Engineering like Software Systems, Software Metrics, Cost Estimation, Neural Networks.

Dr. Simmi Dutta holds a Ph.d Degree from University of Jammu and is presently working as Assistant Professor in the Department of Computer Engineering at GCET, Jammu. She has about 15 years of experience. Her research interests include Software Engineering, Data Engineering and Security, Wireless Networks.