

# An Enhanced Effort Estimation For Software Projects Using Modified COCOMO II With Neural Network

D.Sivakumar<sup>1</sup> and M.Madheswaran<sup>2</sup>

<sup>1</sup> *Department of Information Technology*

<sup>2</sup> *Department of Electronics and Communication Engineering  
Mahendra Engineering College, Mallasamudram, Namakkal, Tamilnadu,  
India-637503 [dsukumar@yahoocse.com](mailto:dsukumar@yahoocse.com), [madheswaran.dr@gmail.com](mailto:madheswaran.dr@gmail.com)*

## Abstract

Software project effort prediction accuracy is improved with modified COCOMO II model and the factors which determine the effort and schedule are presented in this paper. The effort multipliers of COCOMO II model are reduced from 17 to 15 in modified COCOMO II model keeping scale factors unchanged. This model is experimented with the multilayer feed forward neural network using back propagation training algorithm. The percentage of MRE and MMRE are estimated for both the COCOMO II and modified COCOMO II model. It is found from the results that the modified COCOMO II model estimates better performance due to the reduction in number of effort multipliers.

**Index Terms** -- software effort estimation, COCOMO II, back propagation, feed forward neural networks, software cost estimation.

## 1. INTRODUCTION

In the recent years the software project effort estimation is found to be a challenging task due to dynamic change in parameter. In general the software project development focuses on estimation of effort and cost for accurate estimation of software project cost. It is also clear that the accurate estimation will certainly reduce the complexity and hence enhances the project management and decision making. The software project managers estimate the effort and cost to avoid the overestimate and underestimate of effort [1].

COCOMO (**C**onstructive **C**ost **M**odel) is the most widely used algorithmic

cost modeling technique which estimates the effort in person-month for a particular software project in different levels of its development. Complex and nonlinear relationship with measurable function is modeled using artificial neural network [2]. Neural network model is particularly useful if the relationship between input and output is highly complex.

Most of the software effort estimation methods have used the feed forward multilayer perceptron with back propagation learning algorithm. The sigmoid activation function is used in both the hidden and output layers. The artificial neural networks with sigmoid activation function are more sensitive because it may not produce accurate results if some of its parameters are missing. But this activation function has the drawback of slow convergence. Proper selection of neural network pattern and learning rules are highly important in training the neural network and improving its performance [3]. The Performance amplification is achieved by reducing the number of layers and nodes in neural network. The performance of the neural network is improved considering the concise artificial neural network. This method is modeled to fit into modified COCOMO II with linear activation function [4].

M. Shepperd and C. Schofield [5] described an alternative approach to the effort estimation based on the use of analogies sometimes referred to as case based reasoning. The underlying principle is to characterize projects in terms of features that is, the number of interfaces, the development method or the size of the functional requirements. Based on Euclidean distance in n-dimensional space and with n number of project features the similarities in the projects are identified. All the dimensions are standardized in such a way that each dimension may have equal weight. In this work basis for the prediction is the known effort values of the nearest neighbours with the new project.

Detailed review of different studies on the software development effort was provided by Jorgensen [6] with the main goal of contributing and supporting the expert estimation research. Neural networks have learning ability and are good at modeling complex nonlinear relationship, provides more flexibility to integrate expert knowledge into the model. The review results suggest that in some situations expert estimation produces more accurate results than formal estimation techniques. In software project effort estimation the expert estimation is the most frequently used estimation method even though there is no substantial evidence for favouring the use of this model.

Machine learning techniques hold the promise of producing clear and quality output by sensing various inputs in different format. In terms of prediction accuracy the statistical methods based techniques frequently provides low accuracy whereas the hit-or-miss business was always appeared in software cost modeling techniques with better accuracy. Samson [7] applied neural network model, Cerebellar Model Arithmetic Computer (CMAC) to prediction of effort from software code size. CMAC is a perception and function approximation developed by Albus. This neural network was trained on Boehm's COCOMO data set in order to predict effort from size, in the same manner regression techniques were applied for prediction purposes.

Software reliability researches with statistical techniques have proved that the prediction failure occurs due to failure data from similar projects. In this work reliability- growth models are used, in which the reliability has mature enough to guarantee product release. Reliability-growth models shows sign of different prognostic capabilities at diverse testing phases both inside a project and crossway projects. Researchers are finding it practically unfeasible to develop a general model that will afford precise predictions in all situations. Karunanithi [8] produced accurate results by using neural network for estimating software cost. In this work the results heavily relied on training set, this is major set back of this work.

Srinivasan and Fisher [9] reported the use of neural network with back propagation learning algorithm. In this work they express two methods of machine learning, that are indented to be used by the estimators of software project development effort from past data. The experiments designate that these techniques are spirited with conventional estimators on one dataset; they also demonstrated that these methods are perceptive to the facts on which they are trained. Software project effort valuation is supposed to be evaluated by exploring model kindliness on a diversity of historical data. They found that the neural network outperformed other techniques.

The significance in the performance of neural network modeling techniques with complex pattern recognition and nonlinear estimation mechanism has been demonstrated from corner to corner in a remarkable spectrum of applications. Artificial neural network was used to capture the significant attributes of the software development environment to forecast the improvement in prediction accuracy. G.E. Witting [10] examined the performance of back propagation artificial neural networks in estimating software development effort. In this work two different experiments were carried out. Size of software was measured using function point technique and the development effort was measured in terms of hour.

Iman Attarzadeh and Siew Hock Ow [11] suggested that the soft computing approach is best for estimating software project development cost and time using COCOMO II. It has two input cluster from COCOMO II cost drivers, scale factors and one output. Three fuzzy steps are covered by this model; fuzzification process, inference from fuzzy rules and defuzzification process are those steps. Triangular membership function in fuzzy technique is used for software development cost and time estimation and then it's validated with 93 software projects from NASA. Here, the advantages of fuzzy logic and good generalisation are obtained.

COCOMO II model is used in most of the effort prediction purposes, however it has some limitations. Effort calibration of COCOMO II cannot be effectively done with imprecise information. So there is always scope for tuning the COCOMO II calibration models [12] for better predictive accuracy. It shows a promising research direction in software cost estimation; the approaches based on neural network are far from mature.

COCOMO II model parameters are modified by keeping the scale factors remains unchanged and the cost drivers are changed to fifteen drivers. This modified COCOMO II estimation method is modeled with multilayer feed forward neural

network [13]. Iterative processing technique with back propagation learning algorithm is used in this model. The training samples and the test data are identified from the available datasets online. Efforts of software project were estimated and the difference with the actual and the estimated effort are reduced by adjusting the weights of neural network. This process is repeated by fetching the input and outputs with empirical observation, error propagation and weight adjustment in neural network connection until the difference in actual and the estimated effort fall below the tolerable limit [14].

## 2. TUNING COCOMO II WITH ANN

COCOMO [15] model can be used to compute the amount of effort and the time schedule for software projects. COCOMO 81 was an unwavering model in 1980s. The limitation in using COCOMO 81 to the present day effort estimation is that it does not match with the development environment of the late 1990's. So the COCOMO II was published in 1997 and was focused to solve most of the problems associated with recent development environment [16]. This model has three sub models, Application Composition Model, Early Design Model and Post-Architecture Model but they are different from those of COCOMO 81.

Application Composition Model – Suitable for estimating effort and schedule of software projects built with modern GUI-builder tools. In this model effort was estimated based on Object Points. Early Design Model – Rough estimates of a software project's cost and schedules are calculated before determining its entire architecture. It uses a small set of cost drivers, and scale factors. This model is based on unadjusted Function Points and or lines of code. Post-Architecture Model – As the name suggests effort is estimated after the software architecture was well defined. It is a clear and detailed extension of earlier design model. This model is the flanking in structure and formulation to the Intermediate COCOMO '81 and Ada COCOMO models. This uses Source Lines of Code (SLOC) and or Function Points. In this model 17 effort multipliers and 5 scale factors determines the effort and schedule of the software project under development [17]. Cost drivers of COCOMO II model is rated on a scale from Very Low to Extra High as like COCOMO 81. COCOMO II post architecture model is given as:

$$Effort = A \times S^B \times \prod_{i=1}^{17} EM_i \quad (1)$$

Where,

A is the Multiplicative Constant

S is the Size of the software project measured in terms of KSLOC

$$B = 1.01 + 0.01 \times \sum_{j=1}^5 SF_j \quad (2)$$

KSLOC is thousands of Source Lines of Code, Function Points or Object Points. The selection of scale factors (SF) is based on the rationale that they are a significant source of exponential variation on a project effort or productivity variation.

In this work COCOMO II dataset is used for experimentation [18]. Besides, the datasets are publicly available online in the PROMISE repository [19]. The datasets used in this work are COCOMO II dataset and the new project data. There are 63 projects in this COCOMO II data set and are collected from a variety of domains including science, engineering and finance. New project data collected from small size software industry.

In general many of the algorithmic models are not highly precise, because they may reflect the field expert's knowledge [20]. By integrating the algorithmic model with neural network the effective utilization of expert knowledge is achieved. The COCOMO II model uses 17 cost drivers and 5 scale factors. But in this modified COCOMO II the RUSE and the DOCU cost drivers are combined together and represented as DOCU driver. LTEX and TOOL cost drivers are combined and are represented as TOOL cost driver. Table 1 represents the software cost drivers that are used in the modified COCOMO II and the Table 2 represents the scale factors [21]. The major challenge in algorithmic model is the calibration of parameters where influence of parameters such as cost drivers and scale factors are most important in determining the software project effort and cost. In the integrated algorithmic model with neural network the parameters are adjusted through learning [22].

**Table 1 Software Cost Drivers**

Cost Attributes	Description
RELY	Required Software Reliability
DATA	Database Size
CPLX	Product Complexity
DOCU	Documentation and Reusability
TIME	Execution Time
STOR	Main Storage Constraint
PVOL	Platform Volatility
ACAP	Analyst Capability
PCAP	Programmer Capability
PCON	Personal Continuity
APEX	Applications Experience
PLEX	Platform Experience
TOOL	Software and Language Tool Experience
SITE	Multisite Development
SCED	Required Development Schedule

Table: 2. Scale Factors

Scale Factors	Description
PRE	Required Software Reliability
FLEX	Database Size
RESL	Product Complexity
TEAM	Documentation and Reusability
PMAT	Execution Time

Figure 1 shows the simple adaline neural network with multiple inputs. In modified COCOMO II modeling method two adalines are used, that is two multiple input neuron combined to form the madaline, one multiple input neuron for the cost driver and another multiple input neuron for the scale factor [23].

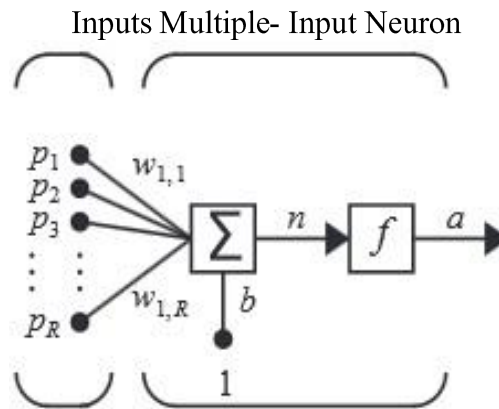


Figure 1 Multiple Input Neuron

Cost driver neuron contains 15 inputs, the scale factor neuron contains 5 inputs and both the neuron adds with the bias. This network consists of two hidden layers that are taken into account in the contribution of effort multipliers and scale factors separately [24]. Sigmoid transfer function has the drawback of slow convergence so the modified COCOMO II model developed with identity transfer function in the input, hidden and output layers.

The initial definition of the modified COCOMO II model with the proposed set of attributes had the following form.

$$Effort = A \times S^{1.01 + \sum_{i=0}^5 SF_i} \times \prod_{i=1}^{15} EM_i \quad (3)$$

Where,

A - Multiplicative constant

- S - Software project size in KSLOC
- SF - Scale Factors
- EM - Effort Multipliers

The modified COCOMO II Model shown in equation “(3)”, is a non linear model. This non linear model is transformed into linear model with logarithms to the base e, and is represented as

$$\ln(PM) = \ln A + \ln(EM_1) + \ln(EM_2) + \dots + \ln(EM_{15}) + [0.01 + SF_1 + \dots + SF_5] \ln(S) \quad (4)$$

The above equation (4) has the form of as given below:

$$O_{est} = [b_1 + u_1 * I_1 + u_2 * I_2 + \dots + u_{15} * I_{15}] + [b_2 + I_{16} + \dots + I_{20}] * [v_i + \ln(s)] \quad (5)$$

Where,

$$O_{est} = \ln(PM);$$

$$I_1 = \ln(EM_1) ; \dots ; I_{15} = \ln(EM_{15});$$

$$I_{16} = SF_1 ; \dots ; I_{20} = SF_5;$$

$b_1$  and  $b_2$  are the biasing factors

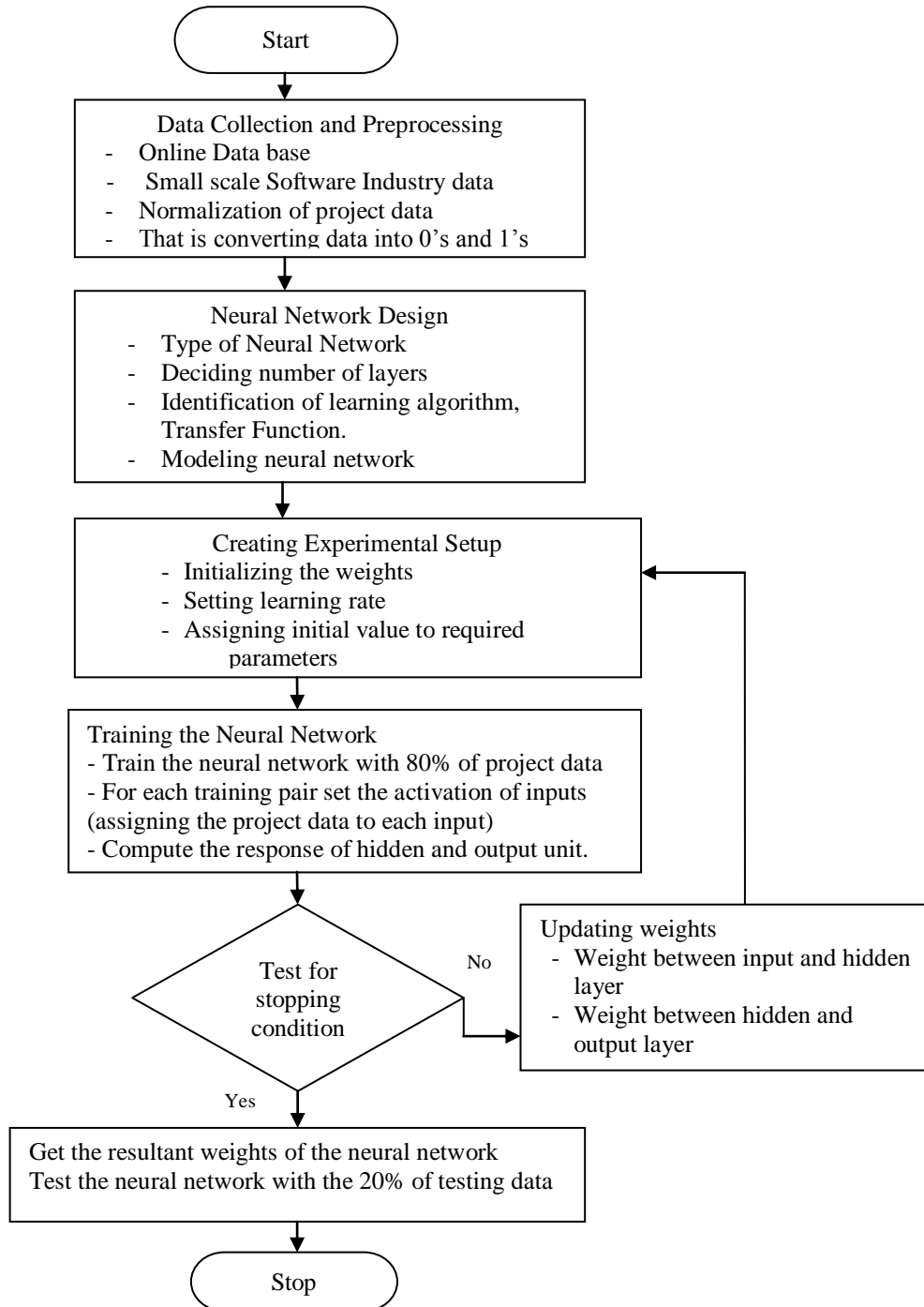
$u_i$  and  $v_i$  are the additional terms introduced in the model to represent the weights.

It is assumed that the initial value of coefficients is to be 1 for  $u_i$  and 0 for  $v_i$  to estimate the output  $O_{est}$ . This estimated effort is compared with that of the actual effort in the logarithmic space. The difference in effort is the error in the estimation, and it should be minimized. It is considered that all the parameters of the modified COCOMO II are responsible for the difference in actual and estimated effort [25].

To incorporate this knowledge with the neural network the difference in effort estimation is propagated to the input factors by adjusting the coefficients. This process is repeated till the difference in actual and estimated efforts to be negligible or the number of iterations exceeds the specific limits.

### 3. METHODOLOGY

Software cost estimation models mainly stands over on cost drivers and scale factors. This model reveals the problem of instability due to the values of the cost drivers and scale factors that affects the sensitivity of effort. Estimation based on historical data is difficult due to inherent complex relationship between the attributes as well as lack of reasoning capabilities. Attributes and relationships used to estimate software development effort could change over time and differ for software development environments. Flow diagram of estimation in modified COCOMO II Model shown in figure 2.



**Figure 2. Flow Diagram of Estimation in Modified COCOMO II Model**

In order to address and overcome these problems actual and estimated efforts of COCOMO II and modified COCOMO II are presented in this section. Even though the methodology and model of this experiment follows the COCOMO II, the main

objective is to increase the prediction accuracy with reduced set of parameters. In all types of empirical software estimation models the estimated cost mainly relies on cost drivers and scale factors. Considering this fact the cost drivers are reduced to 15 attributes. Misjudgment of cost drivers may lead to drastic increase or decrease in effort. With this analysis it is identified that the programmer capabilities, applications experience and software tool experience are the sensitive inputs to the estimation model and requires careful examination while calibrating.

#### 4. RESULTS AND DISCUSSION

In this analysis all the data are normalized to fit in to the logarithmic space. The entire COCOMO 81 dataset is divided into two sets, training set and validation set in the ratio of 4:1. Random selection of 50 project data is utilized for training and 13 project data are used for testing the network. The new project data collected is not utilized in training the neural network and it is tested with the network after training.

Accuracy of modified COCOMO II is evaluated by comparing the actual and the estimated efforts. The evaluation criteria used to assess and compare the performance of the modified COCOMO II is magnitude of relative error (MRE) in addition with the mean magnitude of relative error (MMRE).

$$\%MRE = \frac{|ActualEffort - EstimatedEffort|}{ActualEffort} \times 100 \quad (5)$$

$$\%MMRE = \frac{\sum_{i=1}^N MRE}{N} \quad (6)$$

Where,

N - is the total number of project considered for evaluation.

Formula for calculating the MRE and MMRE is shown in equation (5) and (6). Average of MRE for N projects is referred as MMRE. As a part of evaluation MRE for each project is calculated and the same is compared with the MRE of COCOMO II Model.

Table 3 and 5 shows that the actual and estimated efforts of COCOMO II and a modified COCOMO II model. The statistical value in the test data shows that there is a considerable amount of improvement seen in the modified COCOMO II model when compared with the COCOMO II model. In the project ID 1 the modified COCOMO II model effort 2038 is very close to the actual effort 2040 when compared to the COCOMO II model effort 2018. In the project ID 26 the modified COCOMO II effort 386 is very closer to the actual effort 387 when compared to the COCOMO II model effort 391. This shows a considerable improvement in the effort estimation.

Table 3 summarizes all efforts clearly and it shows better improvement in estimation.

**Table 3. Efforts of COCOMO II Data Set**

S.No	Project ID	ACTUAL EFFORT	Estimated EFFORT	
			COCOMO II	Modified COCOMO II
1	1	2040	2018	2038
2	5	33	39	37
3	9	423	397	424
4	11	218	190	203
5	26	387	391	386
6	34	230	201	235
7	42	45	46	44
8	47	36	33	45
9	50	176	193	163
10	51	122	114	129
11	54	20	24	21
12	56	958	537	956
13	61	50	47	52

**Table 4. %MRE for COCOMO II Dataset**

S.No	Project ID	%MRE of COCOMO II	%MRE of Modified COCOMO II
1	1	1.08	0.1
2	5	18.18	12.12
3	9	6.15	0.24
4	11	12.84	6.88
5	26	1.03	0.26
6	34	12.61	2.17
7	42	2.22	2.22
8	47	8.33	25
9	50	9.66	7.39
10	51	6.56	5.74
11	54	20	5
12	56	43.95	0.21
13	61	6	4
MMRE		11.43	5.49

Table 4 shows the percentage MRE of COCOMO II and modified COCOMO II with COCOMO II dataset. The percentage of MRE is calculated using equation (5). The table 4 shows the %MRE for the entire project considered for testing the model

validity in improving the effort estimation. The % MRE value shows that there is considerable improvement in estimation using the modified COCOMO II. The MMRE of COCOMO II model is 11.43 but the MMRE of modified COCOMO II model is 5.49. It is observed that, the modified COCOMO II model produces the better improvement in estimation.

**Table 5. Efforts of New Project Data**

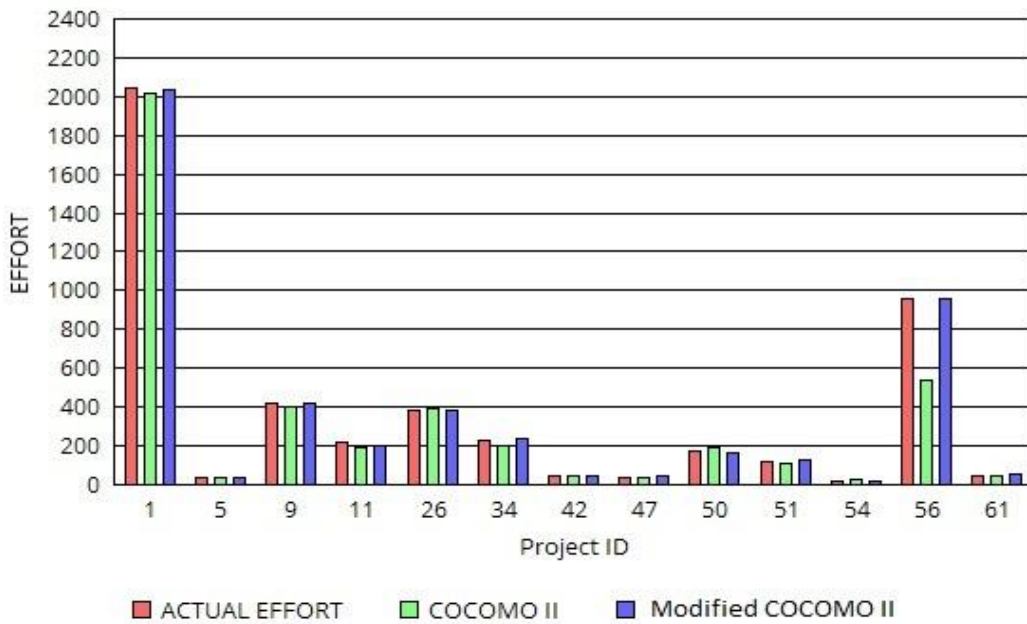
S.No	Project ID	ACTUAL EFFORT	Estimated EFFORT	
			COCOMO II	Modified COCOMO II
1	P1	23	28	24
2	P2	98	113	97
3	P3	10	9	12
4	P4	130	127	136
5	P5	79	81	80
6	P6	86	92	90
7	P7	45	46	44

Table 5 shows the estimated efforts of the project collected for this experiment. The project ID is assigned to the project is P1 to P7. This effort representation table shows that the estimated effort of the modified COCOMO II model is very close to the actual effort when compared to the COCOMO II estimation model.

**Table 6. %MRE for New Project Data**

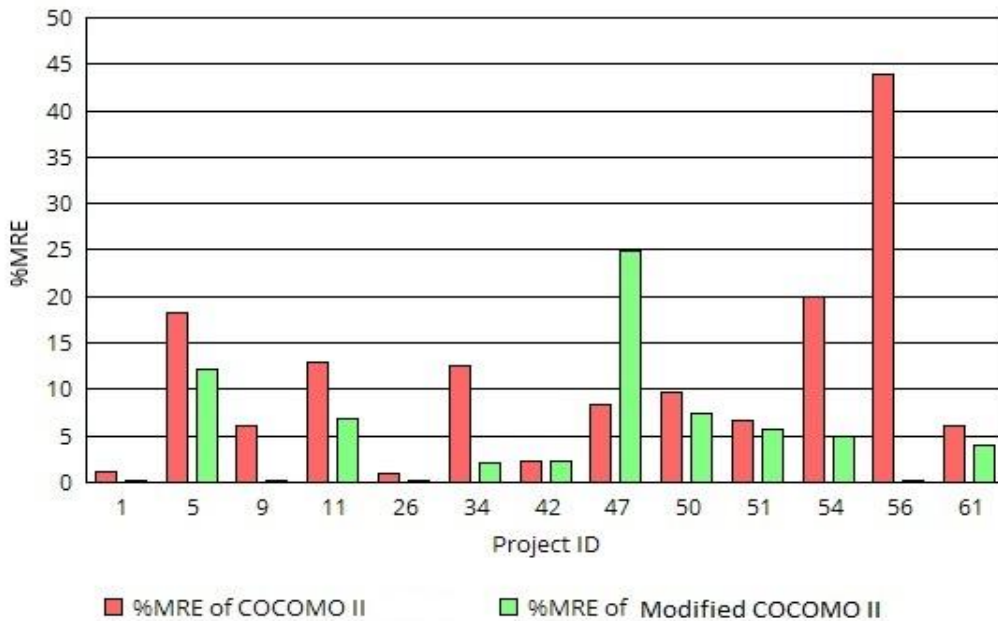
S.No	Project ID	%MRE of COCOMO II	%MRE of Modified COCOMO II
1	P1	21.74	4.35
2	P2	15.31	1.02
3	P3	10	20
4	P4	2.31	4.62
5	P5	2.53	1.27
6	P6	6.98	4.65
7	P7	2.22	2.22
MMRE		8.73	5.45

The percentage of MRE of COCOMO II model and the modified COCOMO II model is shown in table 6. The new project data collected is experimented and the %MRE is evaluated. It is only to justify the model fitness to the recent day project effort estimation. MMRE in COCOMO II model is 8.73 and the same MMRE for the modified COCOMO II model is 5.45.



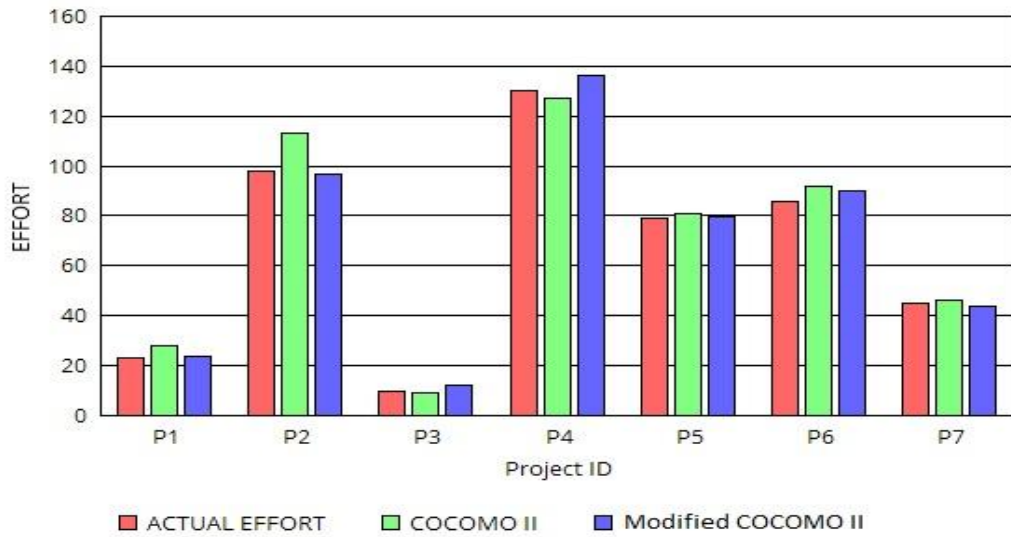
**Figure 3. Effort of COCOMO II dataset**

In figure 3 the estimated and the actual efforts are plotted. It is clearly seen that the effort estimated using modified COCOMO II model is closer to the actual effort when compared to COCOMO II model estimation. This is due to the reduction in redundant parameters which increases the software project effort.



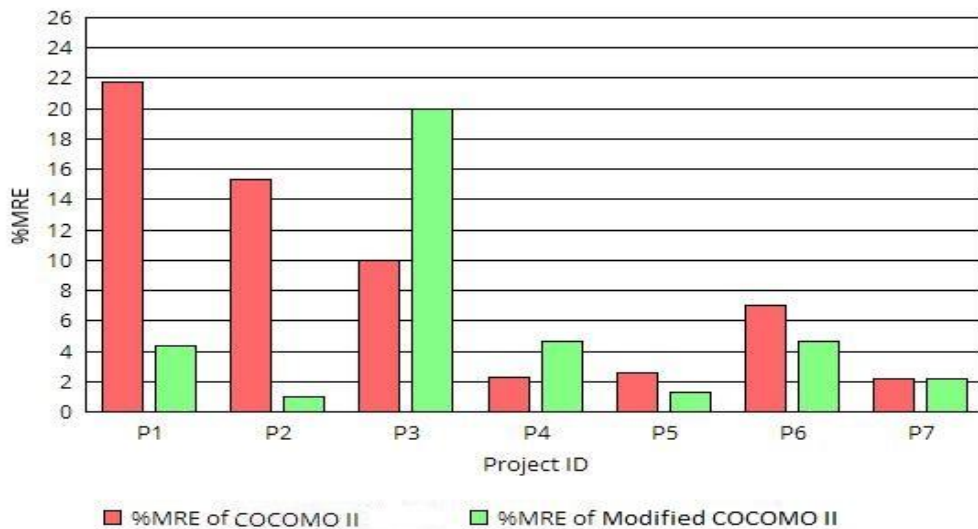
**Figure 4. %MRE of COCOMO II dataset**

The percentage magnitude of relative error for the COCOMO II model and the modified COCOMO II model is given in figure 4. This %MRE representation shows that there is a considerable amount of reduction in estimation error. This reduction in %MRE proves that this modified COCOMO II model improves the accuracy of effort estimation.



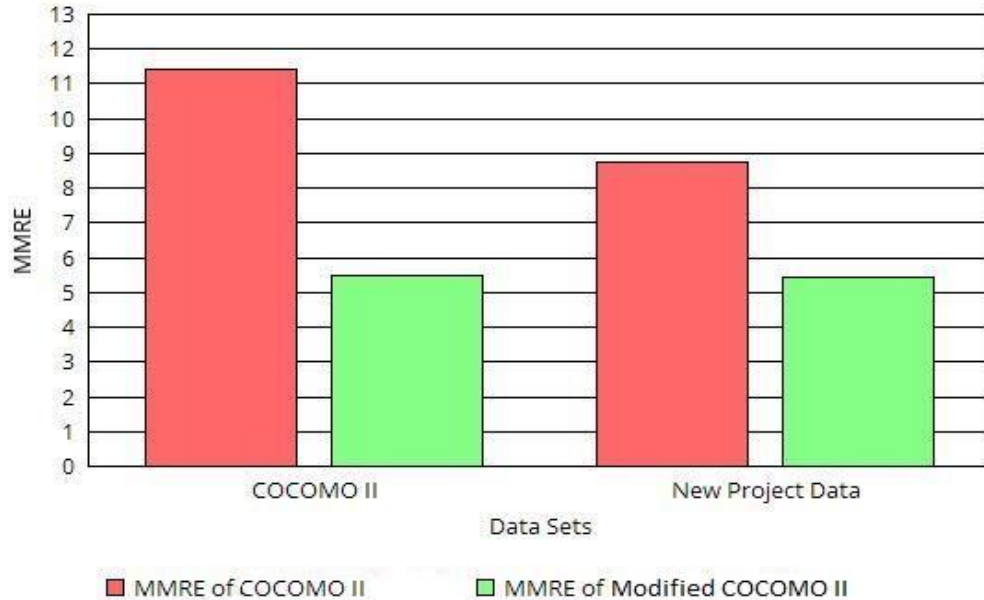
**Figure 5. Effort of New Project data**

Figure 5 depicts the pictorial representation of effort for the project data collected in this experiment. It is also proved that the modified COCOMO II model estimated effort is very near to the actual effort. In this case seven projects are considered for testing and the newly collected project was not utilized for training the neural network.



**Figure 6. %MRE of New Project Data**

MRE value is used to identify the level of improvement in estimation. The figure 6 represents the %MRE of new project data. In this case seven projects are re-considered, even though %MRE of two projects are higher than COCOMO II MRE the mean magnitude of relative error is very less.



**Figure 7. MMRE of COCOMO II dataset and New Project data**

The main objective of the research is to improve prediction accuracy of the model. Figure 7 represents the mean magnitude of relative error in COCOMO II and the modified COCOMO II model. This graphical representation shows one third of improvement in effort estimation.

## 5. CONCLUSION

The cost drivers and the scale factors are found as influencing factors in empirical estimation models. Analyzing the sensitivity level of cost attributes it is identified that the programmer capabilities, applications experience and software tool experience requires at most care while calibrating because it has highest impact on estimating the cost. With the developed modified COCOMO II model, the prediction accuracy is increased considerably. Software project data collected for this experimental analysis is tested with the modified COCOMO II model and it shows remarkable improvement estimation compared to COCOMO II. Mean magnitude of relative error in the modified COCOMO II model clearly shows that the prediction accuracy is improved in the proposed model.

## 6. REFERENCES

1. Boehm, B.W., E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani, and C. Abts, *Software cost estimation with COCOMO II*, Prentice Hall, 2000.
2. Ch.Satyananda., KVSVN Raju., “An Improved Fuzzy Approach for COCOMO’s Effort Estimation Using Gaussian Membership Function”, *Journal of Software*, vol 4, pp 452-459, 2009.
3. Z.Chen, T. Menzies, D. Port, and B. Boehm, “Finding the right data for software cost modeling”, *IEEE Software*, 22 (6), 2005, pp.38-46.
4. Isabelle Guyon, Andr’e Elisseeff, “An Introduction to Variable and Feature Selection”, *Journal of Machine Learning Research*, 3 (2003),pp. 1157-1182
5. M. Shepper and C. Schofield, “Estimating software project effort using analogies,” *IEEE Tran. Software Engineering*, vol. 23, pp. 736–743, 1997.
6. Jørgensen. M., “A Review of Studies on Expert Estimation of Software Development Effort,” *Journal of Systems and Software*, Volume 70, pp. 37-60, 2004.
7. Samson, B., “Software cost estimation using an albus perceptron“, *journal of Info & Softw.*, vol.39, pp.55-60, 1997.
8. Karunanithi, N., "Using neural networks in reliability prediction," *IEEE Software*, pp. 53-59, 1992.
9. Srinivasan, K., Fisher, D., “Machine learning approaches to estimating software development effort,” *IEEE Transactions on Software Engineering*, Volume 21 (2), 126–137, 1995.
10. G.E Witting, “Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort”, *Australian Journal of Info.System*, 1994.
11. Iman Attarzadeh and Siew Hock Ow.” *Soft Computing Approach for Software Cost Estimation*”, *Int.J. of Software Engineering*, *IJSE Vol.3 No.1 January 2010*.
12. Idri, A. Khoshgoftaar, T.M. Abran, A., “Can neural networks be easily interpreted in software cost estimation?” *Proceedings of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'02, Vol.: 2, 1162-1167, 2002*.
13. Abedallah Zaid, Mohd Hasan Selamat, Abdual Azim Abd Ghani, “Issues in Software Cost Estimation”, *IJCSNS International journal of Computer Science and Network Security*, vol. 08, no. 11, pp. 350–356, 2008.
14. J., Hale, A. Parrish, B. Dixon, and R.K. Smith, “Enhancing the Cocomo estimation models”, *IEEE Software*, 17 (6), 2000, pp. 45 – 49.
15. Barry Boehm., Chris Abts., and Sunita Chulani,”*Software development cost estimation approaches – A survey*,” *Annals of Software Engineering*, No.10 pp: 177-205, 2000.
16. Boehm B., Clark B., Horowitz E., Madachy R., "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," *Annals of Software Engineering*, No.1 pp: 57-94, 1995.

17. R.D. Banker, C.F. Kemerer, Scale economies in new software development, *IEEE Transactions on Software Engineering* 15 (10) (1989) 1199–1205.
18. J. S. Shirabad, and T. J. Menzies, “The PROMISE repository of software engineering databases”, School of Information Technology and Engineering, University of Ottawa, Canada, 2005.
19. <http://promise.site.uottawa.ca/SERepository>.
20. Tad Gonsalves, Kei Yamagishi and Kiyoshi Itoh, “Swarm Intelligence in the fine-tuning of Software Development Cost Estimation Models,” Annual IEEE International Computer Software and Applications Conference, pp. 593–598, 2009.
21. M. Hall and G. Holmes, “Benchmarking attribute selection techniques for discrete class data mining,” *IEEE Transactions On Knowledge And Data Engineering*, vol. 15, no. 6, pp. 1437– 1447, 2003.
22. Chris F. Kemerer, (1987) “An Empirical Validation of Software Cost Estimation Models”, *Management Of Computing-Communications of ACM*, Vol:30 No. 5, pp:416-429, May 1987.
23. J.E. Helm, “The viability of using COCOMO in the special application software bidding and estimating process”, *IEEE Transactions on Engineering Management*, 39 (1), 1992, pp. 42 - 58.
24. Barry Boehm., Chris Abts., and Sunita Chulani", Layered Neural Nets for Pattern Recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing* Vol 36, No 7, pp: 1109-1118, July 1988.
25. S. Chulani, B. Boehm, and B. Steece, “Bayesian analysis of empirical software engineering cost models,” *IEEE Transactions on Software Engineering*, vol. 25, July/August 1999.