A Novel Committee Model for Software Effort Estimation Based on Soft Computing Techniques

Pawan Kumar
Department of Computer Sc., D.R.V.D.A.V. Centenary College, Phillaur (Jalandhar)

Abstract

Software development effort estimation is an important activity, which is performed at very early stage of the software development life cycle (SDLC) for business contract generation as well as for resource allocation and planning. The need for accurate and consistently reliable software effort estimation model in software engineering has remained a challenging task. Since the software effort estimation accuracy is directly influenced by the model accuracy, therefore developing good and reliable software effort estimation model is one the most important challenges before the software engineering community. In this area researchers have suggested a variety of models based on various algorithmic and non-algorithmic approaches. The soft computing techniques based on the Fuzzy Logic (FL) and Artificial Neural Network (ANN) seems to prove a good alternative for solving this problem.

This work aims to propose a new software effort estimation model based on the neural networks and fuzzy inference mechanism. This committee model combines the strengths of algorithmic and non-algorithmic effort estimation techniques together in a single framework. In this model, algorithmic model Intermediate COCOMO is used as a base model. Along with the predicted accuracy the proposed model possesses various desirable characteristics like adaptability, interpretability, good generalization and learnability.

KEYWORDS: Effort estimation, COCOMO, Soft Computing, Neuro-fuzzy, Fuzzy Logic (FL), Artificial Neural Network (ANN).

I. INTRODUCTION

Software development effort estimation deals with the prediction of probable time and cost required to complete the specific development task. Generally the software development effort estimations are based on the prediction of size of software, which is a very difficult task to estimate accurately at that point of time. Software development effort estimates at the early stage of the SDLC proves inaccurate because very less and vague information is available about the proposed system [1]. As these initial estimates are very crucial for all the parties involved in the development directly or indirectly, as it supports cost, schedule and resource planning. To the present time many models of software effort estimation have been developed and are in use, which are based on various different theoretical foundations. Most of these models are based on the size (LOC) and Function Point (FP) [5].

Although the task of software development effort estimation at early stages of the development has an uncertain nature and depends on the several factors, which makes it more difficult to formulate mathematically. The accurate and reliable estimation has remained an ongoing challenge in software engineering. Various software cost estimation techniques can be classified as algorithmic and non-algorithmic approaches.

II. EFFORT ESTIMATION APPROACHES

Software effort estimation is the oldest and most important aspect of software metrics towards rigorous software measurement. Extensive research had been carried out, to come up with different effort estimation models having different theoretical foundations. This section

discusses the evolution of some important algorithmic and non-algorithmic estimation techniques overtime.

A. Estimation Techniques Based on Algorithmic Approach:

Some of the popular algorithmic models are Bohem's COCOMO'81, COCOMO II and Putnam's SLIM. All such models require inputs, such as size (in terms of lines of code), number of reports, number of user screens, complexity and other technical aspects which influence the effort estimation process. The inputs required for these models are not easy to acquire accurately at the early stages of the development life cycle. As the basis of these models is historical data, so understanding and calculations of these models is very difficult due to the inherent non-linear and complex relationship between the contributing factors [9]. Models based on algorithmic approach lacks in adaptability as well as reasoning capabilities. These weaknesses of the algorithmic models motivated the researchers to explore the non-algorithmic techniques based on soft-computing.

COCOMO developed by Boehm while at TRW can be used in three phases. Basic COCOMO has two inputs: mode and size (in thousands of delivered source code), and it produces a nominal effort estimate using the equation [1].

$$\text{Effort (nominal)} = a\,(\text{KLOC})^{b}$$

The mode attempts to characterize system complexity, overall management, and tools being used. The original COCOMO model (COCOMO'81) distinguishes between three basic development modes: *organic*, *semidetached*, and *embedded*. This classification is commonly used throughout the software engineering community.

TABLE I : Mode Coefficient and Exponents

Mode	A	B
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.2

The intermediate COCOMO is an extension of the basic COCOMO model, it uses the product 15 cost drivers and nominal effort (from Basic COCOMO) to compute the adjusted nominal effort estimate.

$$\text{Effort} = a(\text{KLOC})^{b} \times \text{EAF}$$

EAF (Effort Adjustment Factor) is product of 15 project attributes.

Putnam also developed an early model known as SLIM in 1978. Both these models make use of data from past projects and are based on linear regression techniques, take number of lines of code as the major input to their models. A survey on these algorithmic models and other cost estimation approaches is presented by Boehm et. al.[7]. Algorithmic models such as COCOMO are unable to present suitable solutions that take into consideration technological advancements. This is because these models are often unable to capture the complex non-linear relationships (effect of each factor in a model to the overall prediction made using the model) that are evident in many software development environments. They can be successful within a particular type of environment, but not flexible enough to adapt to a new environment. They cannot handle categorical data and lack in reasoning capabilities. These weaknesses of algorithmic models are

the motivation for the number of studies exploring the alternate non-algorithmic methods (e.g. Fuzzy Inference, Neural Network etc.) for effort estimation.

B. Estimation Techniques Based on Non-algorithmic Approach:

The new computation technique for the effort estimation that are non-algorithmic viz. a set of approaches that are soft computing-based emerged in the 1990s and turned the attention of researchers towards them. This section discusses some of the non-algorithmic models, that are soft computing-based. Soft computing encompasses methodologies centering in fuzzy logic (FL), artificial neural networks (ANN) and evolutionary computation (EC). These methodologies handle real life ambiguous situations by providing flexible information processing capabilities and support learning.

Many researchers have contributed towards software development effort prediction using soft computing techniques which handle the imprecision and uncertainty in data aptly due to their inherent nature.

1) Fuzzy Logic Based Approach: Another good alternative approach for the effort estimation is use fuzzy inference system which does mapping of linguistic terms such as "small size" and "high complexity" attached with the variables. A general fuzzy system has three components. Input Fuzzifier for the fuzzification of various cost drivers, inference engine having a rule base and an output defuzzifier for the defuzzification of the final output (effort). The inputs/output of fuzzy logic based effort estimation system can be either linguistic or numeric. The fuzzy membership functions are defined by the domain experts by using the historical data. A fuzzy rule bank is also defined by the experts, which is a series of If-then rules. These rules are executed in parallel when the certain inputs are applied to the systems.

2) Artificial Neural Network (ANN) Based Approach: Significant efforts have been put into the research of developing software effort estimation model based on artificial neural network because of their highly useful features. Neural networks are good at complex non-linear mapping and are based on the principle of learning by examples.

A neural network is a collection of highly interconnected processing elements which mimic human brain in structure and knowledge representation. The main idea of using such technology for effort estimation is their capability to perform non-linear and sophisticated computations in a similar manner to the biological neurons in the human brain. Neural networks are being characterized in terms of three entities: neurons, interconnection structure and learning algorithms. Most of the software effort estimation models developed using the neural networks use multilayered feed-forward networks. The development of such a neural network model starts with an appropriate layout of neurons, the number of neurons in the each layer and the manner in which these are connected. The activation function of the nodes and the specific training algorithm to be used must also be determined. Once the network has been built, the model must be trained by providing it with a set of historical project data as input and the corresponding known actual values for the project effort. The model then iterates on its training algorithm, automatically adjusting the weights until the model weights converge. Once the training is complete and the appropriate weights for the network links are determined, new input can be presented to the neural network to predict the corresponding project efforts.

III A NOVEL NEURO-FUZZY COMMITTEE MODEL

In the areas where any single technology is not having all the desired features, constituting an expert committee of two or more such technologies which complement each other is a good idea, which is basis of this proposed model. As the COCOCMO model is very simple,

understandable and easy to use technique of effort estimation, but lacks in adaptability. Fuzzy Logic is very efficient at handling imprecise and vague inputs, expert knowledge can be incorporated in the forms of rules suggested by the domain experts, reasoning process is understandable, and rules can be updated or deleted very easily when required.

On the other hand Artificial Neural Networks are good in mapping non-linear relationship between the inputs and outputs. These are having very good learning and generalization capabilities. So combing the strengths of these different technologies into a single model can produce a highly accurate, reliable and consistent model of software effort estimation.

Our novel neuro-fuzzy committee model for the software development effort estimation is shown in Fig1. It is an open model, other technologies having the desired characteristics (adaptability, understandability, complex non-linear mapping, support learning) can be made member of this estimation committee.

The input for this model is the estimated size, mode and 15 cost drivers. The output produced by the model is estimation of efforts required for the development of the proposed system.

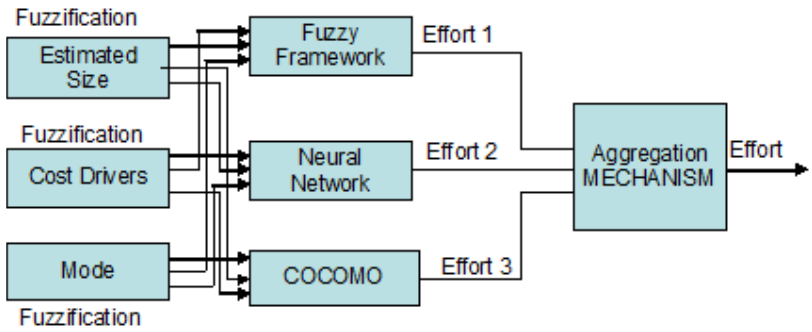

Fig. 1: A neuro-fuzzy committee model for software effort estimation.

A. Model Description and Implementation

This committee model is having three main components (Algorithmic component COCOMO, Neural network component and fuzzy inference component) and one aggregation mechanism for combining the results of three components and generating the estimated efforts. The NASA public dataset of 92 completed projects is used in the implementation and validation of the model.

1) Algorithmic Component (COCOMO): The intermediate COCOMO which is an extension of the basic COCOMO model, is used to implement the algorithmic component of the model. It uses the product 15 cost drivers and nominal effort (from Basic COCOMO) to compute the comprehensive effort estimate.

$\text{Effort} = a(\text{KLOC})^b \times \text{EAF}$

a , b are mode coefficient and exponent respectively taken according to table1.

EAF (Effort Adjustment Factor) is product of 15 project cost drivers.

2) Fuzzy Inference Component: The fuzzy inference component of the model is implemented by using MATLAB's FIS by using the intermediate COCOMO as a base model. The inputs of this component are divided into two groups. First group consists of size and mode which are required to compute the nominal effort part of the output. Second group contains

rating scales of the 15 cost drivers and corresponding effort multipliers as suggested by Barry Boehm [3]. Fuzzification of these two parts (nominal effort and EAF) of COCOMO is done to capture the vagueness in the inputs in an organized manner [5]. Expert's knowledge is incorporated into the fuzzy component by developing a rule base, which is understandable and modifiable. The first input group when fed to FIS (fuzzy inference system) produces crisp nominal effort. The second input group of 15 independent cost drivers are fuzzified independently and are used to produce the crisp EAF. To produce the crisp effort product of crisp nominal effort and crisp EAF is performed. The fuzzification of size and mode used to infer the nominal effort is shown in figure 2. The fuzzy sets of cost driver (programmer capability and its impact on the efforts is shown in figure 3.

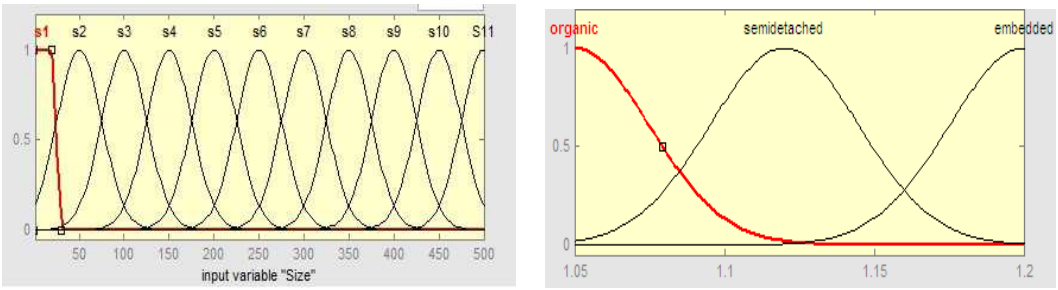

Fig. 2: Input variable "Size" and "Mode" represented as Gaussian MFs

TABLE II: THE PCAP COST DRIVER RANGE DEFINITION IN TERMS OF PERCENTILES

Very Low	Low	Nominal	High	Very High
15	35	55	75	90

TABLE III: THE PCAP EFFORT MULTIPLIER RANGE DEFINITION

Very Low	Low	Nominal	High	Very High
1.42	1.17	1.0	0.86	0.70

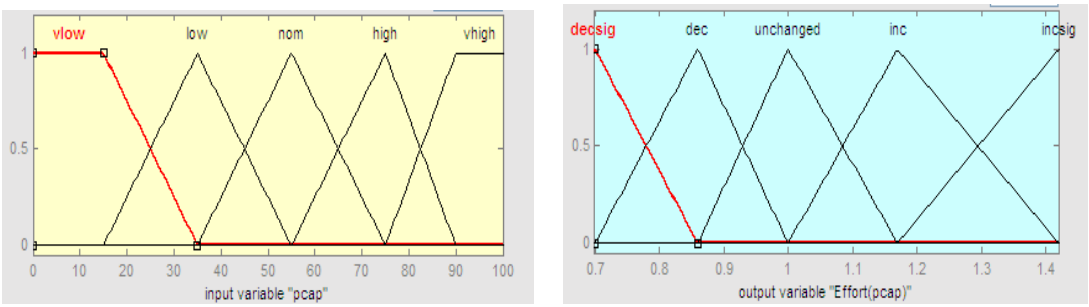

Fig. 3: Membership functions for the Cost Driver "pcap" and its effect on efforts.

Rules formulated for the cost driver programmer capability based on the defined fuzzy sets are of following form:

IF pcap is vlow then Effort(pcap) incsig.
IF pcap is low then Effort(pcap) inc.

IF pcap is nom then Effort(pcap) unchanged.
IF pcap is high then Effort(pcap) dec.
IF pcap is vhigh then Effort(pcap) decsig.

3) Neural Network Component: A feedforward backpropagation neural network with 18 neurons in the input layer, one neuron in the output layer and having a single hidden layer is created with newff function of MATLAB Neural Toolbox. A subset of 45 projects is selected for training the network. The learning performance of the neural network component over the 1000 epochs is shown in figure 4.

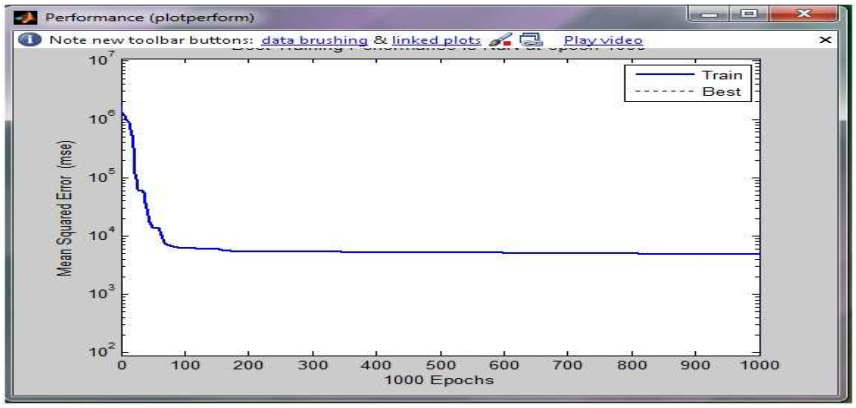

Fig. 4: Learning performance of the Neural Network component

B. Experimental Results and Analysis

The prediction accuracy of the developed model is compared with COCOMO in terms of

- Magnitude of Relative Errors (MRE) in Comprehensive Effort
- Mean Magnitude of Relative Errors (MMRE) in Comprehensive Effort

MRE = |actual effort − estimated effort| / |actual effort|

MMRE=1/n $\sum MRE_i$

The following graph shows the comparison of magnitude of relative error (MRE) over 92 projects from the NASA dataset by COCOMO, Fuzzy inference system, Neural network and the proposed model. Table 4 shows the MRE produced by different approaches.

TABLE IV: MRE of VARIOUS EFFORT ESTIMATION APPROACHES

COCOMO	Fuzzy Inference System	Neural Network	Proposed Model
18.4981281	27.8069	5.239355025	17.1815

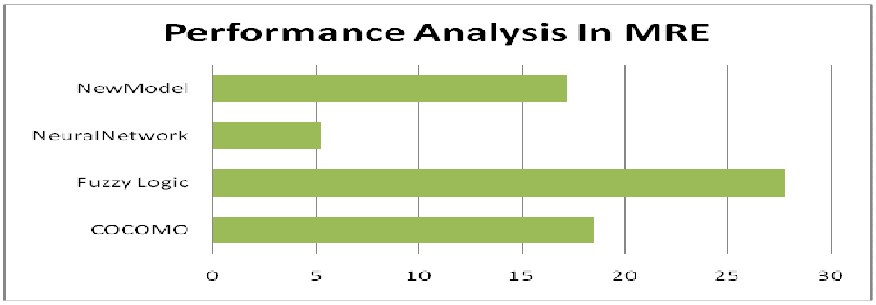

Fig. 5: MRE of various approaches used in effort estimation

The graph given under shows the comparison of mean magnitude of relative error (MMRE) over 92 projects from the NASA dataset by COCOMO, Fuzzy inference system, Neural network and the proposed model. Table 5 shows the MMRE produced by different approaches.

TABLE V: MMRE of VARIOUS EFFORT ESTIMATION APPROACHES

COCOMO	Fuzzy Inference System	Neural Network	Proposed Model
0.201067	0.30225	0.05695	0.18275

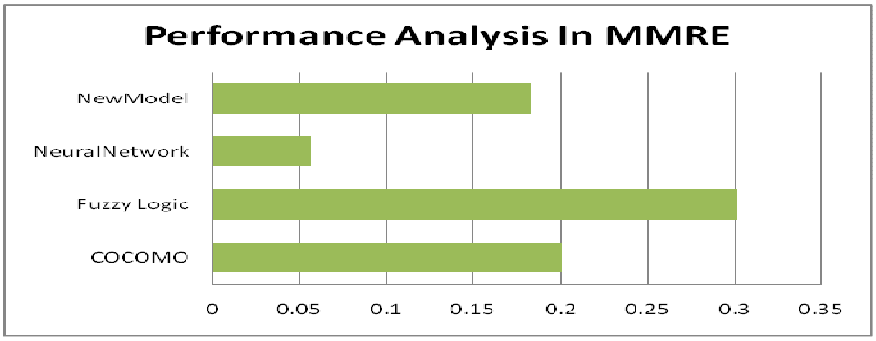

Fig. 6: MMRE of various approaches used in effort estimation

IV. CONCLUSIONS

The study has presented a more accurate and transparent fuzzy logic and neural network based framework, for software development effort prediction. The framework is described, implemented and validated. The study has demonstrated the capabilities of the framework through validation carried out on NASA public dataset of completed projects.

In conclusion, although the proposed model offers prediction accuracy close to the COCOMO model when developed using artificial datasets, it has the potential to offer better prediction accuracy than the COCOMO model when augmented using the industry data and expert knowledge. In short, the developed framework is more transparent, visual, adaptable to changing environments, capable of handling categorical values, resistant to different results for similar projects, requires less data and most importantly tolerant to imprecision.

REFERENCES

[1]Roger S. Pressman and Bruce R. Maxim,Software Engineering-A Practitioner's Approach, 7 ed., McGraw-Hill Higher Education,2009

[2] K. Strike, K. El-Emam, and N. Madhavji, "*Software Cost Estimation with Incomplete Data*", IEEE Transactions on Software Engineering,27(10) 2001.

[3] A. C. Hodgkinson, and P. W. Garratt, "*A Neurofuzzy Cost Estimator*", in: Proceedings of the Third International Conference on Software Engineering and Applications—SAE 1999, pp. 401–406.

[4] C. Kirsopp, M. J. Shepperd, and J. Hart, "*Search Heuristics, Case-Based Reasoning and Software Project Effort Prediction*", Genetic and Evolutionary Computation Conference (GECCO), New York,AAAI, 2002.

[5] J. Ryder, "*Fuzzy Modeling of Software Effort Prediction*",Proceedings of IEEE Information Technology Conference, Syracuse, NY, 1998.

[6] Z. Fei, and X. Liu, "*F-COCOMO: Fuzzy Constructive Cost Model in Software Engineering*", Proceedings of the IEEE International Conference on Fuzzy Systems, IEEE Press, New York, 1992 pp 331-337

[7] H.K. Verma and V. Sharma. "*Handling Imprecision in Inputs Using Fuzzy Logic to Predict Effort in Software Development*" Proceedings of IEEE International Advanced Computing Conference, ISBN: 978-1-4244-4790-9, March 2010.

[8] M. A. Ahmed, M. O. Saliu, and J. AlGhamdi, "*Adaptive Fuzzy Logic-based Framework for Software Development Effort Prediction*", Information and Software Technology Journal, 47 2005, pp. 31-48.

[9] C. Kirsopp, and M. J. Shepperd, "*Making Inferences with Small Numbers of Training Sets*", Sixth International Conference on Empirical Assessment & Evaluation in Software Engineering, Keele University, Staffordshire, UK, 2002.

[10] A. Idri, and A. Abran, "*COCOMO Cost Model Using Fuzzy Logic*", Seventh International Conference on Fuzzy Theory and Technology, Atlantic City, NJ, 2000.

[11] M. Shepperd, and G. Kadoda," *Comparing Software Prediction Techniques Using Simulation*", IEEE Transactions on Software Engineering, 27(11) 2001, pp. 1014–1022.

[12] M. A. Ahmed, and M. O. Saliu, "*Soft Computing Based Effort Prediction Systems—A Survey*", in:E. Damiani, L.C. Jain (Eds.), Computational Intelligence in Software Engineering, Springer-Verlag, July 2004.

[13] B. Boehm, "*Software Engineering Economics*", Prentice-Hall, Englewood Cliffs, NJ, 1981.

[14] B. K. Clark, "*The Effects of Software Process Maturity on Software Development Effort*", PhD Dissertation, Faculty of Graduate School, University of Southern California, 1997.

[15] W. Pedrycz, H. F. Peters, and S. Ramanna, "*A Fuzzy Set Approach to Cost Estimation of Software Projects*", Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, Alberta, Canada, 1999.