Software Cost Estimation Using Neuro Fuzzy Logic Framework

A. Mary Christina[1], V. Banumathy[2]

[1]*Lecturer, Department of Computer Engineering, Srinivasa Subbaraya Govt. Polytechnic college, Puthur, India*

[2]*Lecturer, Dept. of Computer Engineering, A. D. J. Dharmambal Polytechnic College, Nagapattinam, India*

***Abstract*: Software cost estimation is one of the most demanding tasks in software engineering. Software organizations /have a huge necessitate for estimating software projects during the early stages of software development lifecycle in order to handle their resources such as money, manpower, etc. The software effort estimation model like Constructive Cost Model (COCOMO) is a mathematical algorithmic model that estimates efforts relying upon the accurate estimation of size or complexity. The precision of algorithmic models for software cost prediction is limited due to their inability to maintain imprecision and uncertainties related with the software project attributes like size, programmer experience, etc. This work improves the accuracy and sensitivity of one of a broadly used model COCOMO-II by incorporating a neuro-fuzzy component into the model. Neuro-Fuzzy models are the combination of Artificial Neural Network and Fuzzy Logic. Artificial Neural Network has the capability to learn from previous data. It model difficult relationships between both independent variables and dependent variables. Our proposed work is compared with neural network approach and fuzzy logic approach based on the value of MMRE and PRED. We used MATLAB to assess these neural, fuzzy logic and neuro-fuzzy systems.**

***Keywords*: Effort Estimation, Neuro-Fuzzy Logic, COCOMO-II**

1. Introduction

Software cost estimation has gained great importance in the last two decades due to its essential necessity for efficient effort estimation in software analysis. Software cost estimation refers to the estimation of the human effort and time required to develop a software artifact. The accurate estimation of the development effort and cost of a software system is one of the significant and challenging tasks for software project management. It is very useful in contract negotiations, project scheduling and capable allocation of resources. Yet, estimates at the preliminary stages of the project are the most intricate to obtain because the primary source to estimate the cost comes from the requirement specification documents .Enhancing the estimation techniques that are currently obtainable to project managers would facilitate improved control of time and money in software development. Furthermore, any improvement in the accuracy of estimating the development effort can significantly minimize the costs from errors, such as estimating incorrectly, ambiguous tendering bids, and disabling the monitoring progress [1], [2]. In the past decades, some significant software estimation algorithmic models have been published by researchers, as Constructive Cost Model (COCOMO) (Boehm et al. 2000), and Function Points (Albrecht 1979; Jones 1998). Model-based techniques have numerous strengths, the most important of which are objectivity, repeatability, the occurrence of supporting sensitivity analysis, and the capability to calibrate to previous experience (Boehm 1981). On the other hand, these models also have some drawbacks. One of the drawbacks of algorithmic models is their lack of flexibility in adapting to new situation. The new development environment generally entails a unique situation, resulting in inaccurate inputs for evaluation by an algorithmic model. As a fast changing business, the software industry often faces the problem of instability and hence algorithmic models can be rapidly outdated. The outputs of algorithmic models are depends on the inputs of size and the ratings of factors or variables (Boehm 1981). Hence, erroneous inputs to such models, resulting from obsolete information, cause the evaluation to be inaccurate. Another disadvantage of algorithmic models is the strong collinearity among parameters and the difficult non-linear relationships between the outputs and the contributing factors.

In this work, we propose a novel neuro-fuzzy COCOMO-II model to estimate software development effort by combining neurouzzy technique with the accepted COCOMO-II model. When we join the neuro-fuzzy approach with the standard COCOMO models, we can take advantage of some attractive features of a neuro-fuzzy approach such as its learning/adaptation ability and good interpretability. Consequently, our model is potential of generalization, an essential criterion for successful applications of fuzzy logic and neural networks techniques. Another feature of our model is that it allows for continuous rating values as input, which discards the problem of similar projects with large different cost estimations. The Neuro-fuzzy technique permits the integration of numerical data and expert knowledge and can be a powerful tool when tackling significant problems in software engineering.

2. Related work

Neha Sharma et. al. (2013) proposed model to evaluate parameters for NASA software project dataset using a genetic algorithm (GA). The experimental result shows that the three models (Organic model, Semi-detached model and Embedded model) takes a lot of time and performs inferior than proposed model [3].

Swarup Kumar et. al. (2011) proposed fuzzy software cost estimation model that handles vagueness obscurity and a comparison is made with other famous software cost estimation models. Fuzzy logic method is used to address the difficulty of obscurity and vagueness exists in software effort drivers to estimate software effort [4]. A.Ahmed et al. [5] have presented about the adaptive Fuzzy Logic framework for effort prediction as algorithmic effort prediction models are not very competent to cope with the uncertainty and ambiguity present in the software projects. As a result the training and adaptation algorithm used in the framework are capable to handle the imprecision, explain the prediction via rules, provides transparency in the prediction system, and could adapt to the new environment when new data is obtainable. This work includes the transparent FL-based framework which allows contribution from experts, and is set with training and adaptation algorithm for development effort prediction. This work demonstrates the capabilities of the framework via empirical validation carried out on artificial datasets and the COCOMO.

Martin et al. [6] have investigated the contrast between Fuzzy Logic Models (FLM) and Linear Regression Model (LRM) because the engineers have the less capability which is being offered by personal training, therefore they cannot support their teams to generate reliable results. This work includes the evaluation criterion which is based on the magnitude of error relative to the estimate (MER) in addition to the mean of MER (MMER). In this work, small programs are being developed by programmers. Along with these programs, Fuzzy Logic Models were generated to calculate the effort. Confirmation and Validation of the models are done. The output thus generated by the Fuzzy Logic model and Linear Regression produce the similar predictive accuracy, so Fuzzy Logic Model can be used as a substitute to estimate effort. Iman Attarzadeh et. al. (2010) proposed a model for managing imprecision and uncertainty by using the fuzzy logic systems. The proposed fuzzy logic model shows well software effort estimate evaluation criteria as compared to the traditional COCOMO. The experimental results demonstrate that applying fuzzy logic technique to the software effort estimation is a possible approach to addressing the problem of uncertainty and vagueness [20]. Gharehchopogh (2011) did a case study for software cost estimation using Neural Network (NN) architecture for finding necessary effort of new software. The results indicate that the NN model provides the very best algorithmic method to predict and estimate software costs [7].

Wei Lin Du et al. [8] proposed an approach combining the neuro-fuzzy technique and the SEER-SEM effort evaluation algorithm. The continuous rating values and linguistic values are the inputs of the proposed model for avoiding the deviation in estimation among related projects. The performance of the proposed integrated method has been optimized by designing and calculated with the data published in the historical projects. The evaluated results specify that the estimation with the proposed fuzzy model containing analogy reasoning make better results in comparison with the existing techniques [9] that uses feature selection algorithm. Xishi Huang et al. [10] also developed neuro-fuzzy Constructive Cost Model (COCOMO) for software cost estimation which uses the attractive features of a neurofuzzy approach, like learning ability and good interpretability, in COCOMO model. Chen et al. [19] concluded that the COCOMO II model can be enhanced via WRAPPER feature subset selection method developed by the data mining community. Using data sets from the PROMISE storage area, they showed WRAPPER considerably and dramatically enhances COCOMO II's predictive power.

3. Neural network models

Significant effort has been put into the research of increasing software estimation models using neural networks [11, 12, and 13]. Neural networks are based on the principle of learning from example with no previous information being specified. Neural networks are characterized based on three entities such as neurons, interconnection structure and learning algorithms. Majority of the software models developed using neural networks utilize multilayer feed-forward networks. The improvement of such a neural network model starts with a suitable layout of neurons, or connections between network nodes. This involves defining the number of layers of neurons, the number of neurons within each layer, and the manner in which they are connected. The activation functions of the nodes and the specific training algorithm to be utilized must also be found out. Once the network has been built, the model must be trained by offering it with a set of historical project data input and the consequent known actual values for project effort. The model then iterates on its training algorithm, routinely adjusting the weights until the model weights converge. Once the working out is complete and the suitable weights for the network links are determined, new input can be offered to the neural network to estimate the consequent project effort. In general, large data sets are required to accurately train neural networks.

4. Fuzzy logic models

A fuzzy method is a mapping between linguistic terms, such as medium complexity and high cost that are attached to variables. Thus an input into a fuzzy method can be either numerical or linguistic with the same applying to the output. A classic fuzzy system is made up of three major components such as fuzzifier, fuzzy inference engine and defuzzifier. The fuzzifier transforms the input to linguistic terms using membership functions that signify how much a given numerical value of some variable fits the linguistic term being considered. The fuzzy inference engine performs the mapping among the input membership functions and the output membership functions using fuzzy rules that can be get from expert knowledge of the relationships being modeled. The larger the input membership degree, the stronger the rule fires, thus the stronger the pull to the direction of output membership function. Since numerous different output membership functions can be contained in the corresponding rules triggered, a defuzzifier carries out the defuzzification process to merge the output into a single label or numerical value as needed.

5. COCOMO framework

COCOMO (Constructive Cost Model) is the finest known algorithmic cost model published by Barry Boehm in 1981. It was developed from the study of sixty three software projects. The COCOMO model is a hierarchy of software cost estimation models.

A. Basic COCOMO Model

Fundamental COCOMO computes software development effort as a function of program size. Program size is expressed in predicted thousands of source lines of code (SLOC). COCOMO [10] applies to three classes of software projects: In organic mode simple projects that engage small teams functioning in well-known and stable environments. In Semi-detached mode projects that engage teams with an assortment of experience. It is in between organic mode and embedded mode. In embedded mode complex projects that are developed under rigid constraints with changing requirements. The basic COCOMO equations are,

Effort Applied, E = a × (SLOC) b [man-months]
Development Time, D = c × (Effort Applied) d [months]
People required, [count]

Where, SLOC is the identify number of delivered lines (expressed in thousands) of code for project, here the coefficients a, b, c and d are depends upon the three modes of improvement of projects.

6. COCOMO II and neuro fuzzy logic framework

A. COCOMO II model

It is degeneration based software cost estimation model and thought to be the majority cited, best known and the most plausible of all conventional cost prediction models.

COCOMO II consists of the following models [14]:

B. Application composition model

It assumes that systems are formed from reusable components, scripting or database programming. It involves prototyping efforts to solve potential high-risk issues such as user interfaces, software/system communication, performance, or technology maturity. It is used during the premature stages of development when prototype of user interface is obtainable. Software size estimates are based on application points / object points, and a simple size/productivity formula is utilized to estimate the effort required.

C. Early design model

To acquire rough estimates of a project's cost and duration prior to determined its whole architecture. It uses a small set of new cost drivers and new calculating equations. It utilizes Unadjusted Function Points (UFP) as the measure of size.

D. Post architecture model

On one occasion the system architecture has been designed, a more precise estimate of the software size can be made. It includes the actual development and maintenance of a software product. One could use function points or LOC as size estimates with this model. COCOMO II model describes various cost drivers that are used in the Post Architecture model. The cost drivers for COCOMO II are rated on a scale starting from Very Low to Extra High.

E. Effort estimation

In COCOMO II model effort is expressed as Person-Months (PM). A person month is the amount of time one person spends functioning on the software development project for one month. COCOMO II handles the number of person-hours per person-month, as an adjustable factor with a nominal value of 152 hours per Person-Month. This number excludes time usually devoted to vacations, holidays, and weekend time off. The number of person-months is diverse from the time it will take the project to finish. This is called the development schedule. The inputs to the model are the Size of software development, an exponent, E, a constant, A, and a number of values called effort multipliers (EM). The number of effort multipliers based on the model.

F. Requirement for effort estimation

- It would provide increased control of time and overall cost profit in software development life cycle.
- Software development effort estimates are the origin for project bidding and planning.
- Generally software effort estimation has even been recognized as one of the three most demanding challenges in software application areas. In the development process, the cost and time estimates are helpful for the preliminary rough validation and monitoring of the project’s completion process. And in addition, these estimates may be helpful for project productivity assessment phases.

G. Neuro fuzzy logic

Neural networks and fuzzy logic combination is the general idea behind the neuro-fuzzy system. NFS is a fuzzy system collectively with neural networks to increase some characteristics like flexibility and adaptability [16], [17], [18]. This work uses the second approach. There are 3 main components in the intelligent model:

H. Neuro-fuzzy inference system (NFIS)

It manages the dependencies among cost drivers. The inputs are the ratings of the cost drivers, and its output is the tuned ratings of cost drivers.

I. *Neuro-fuzzy subsystem (NFi)*

The input is the adjusted rating of the ith cost driver and the output is the multiplier rate of the ith cost driver that is used for input of the COCOMO. There are 22 cost drivers in our model. Each cost driver represents one factor of the development effort, such as product difficulty, applications experience etc. To estimate the contribution, we use six qualitative rating levels. When defined in linguistic terms, the six rating levels are Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH) and Extra High (XH). Each rating level of each cost driver relates to a quantitative value utilized in the COCOMO model. Sub-model NFi is used to convert this rating of a cost driver into a quantitative multiplier value and to calibrate these relations by industry project data. In this work, we define a fuzzy set for each linguistic term of every cost driver are, very low, low, nominal, high, very high, extra high. We permit the membership functions are triangular functions or other functions, and prefer the universe of discourse to be the interval [1, 6]. We use fuzzy numbers "about 1" to about 6" to represent linguistic terms very low, low, nominal, high, very high, extra high, respectively.

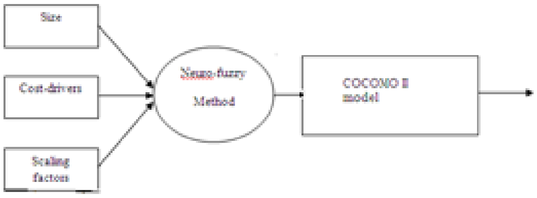

Fig. 1. Proposed Architecture

COCOMO II consists of size, cost drivers and scale factors input and effort as output which is measured in person months (PM).The issue with software effort estimation is that it mostly depends upon single values of size, scale factors and cost drivers. The size of the project is estimated depends upon previously completed projects that are somewhat related with the current project. Also cost drivers and scale factors required to have through assessment rather than assigning a fixed numeric value. In the proposed model COCOMO II's input parameters are size, cost drivers and scale factors are taken into consideration. This novel framework has attractive attributes, chiefly the fact that it can be generalized to several different situations and can be used to produce more specific models. In actual fact, its generalization is one of the purposes of scheming this framework. Its implementation is not limited to any particular software estimation model. The algorithmic model in the framework can be one of the current popular algorithmic models such as COCOMO. When various algorithmic models are implemented into this framework, the inputs and the non-rating values are dissimilar.

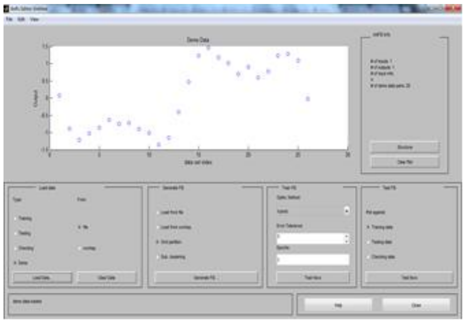

Fig. 2. Neuro-fuzzy model

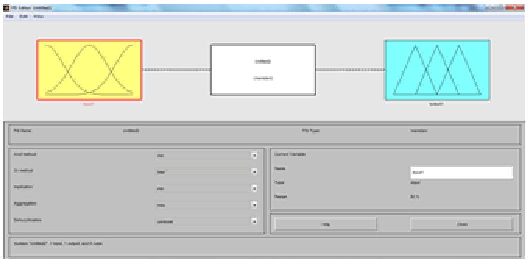

Fig. 3. FIS Editor

J. *Performance evaluation*

Many criteria to assess and compare effort estimation models are proposed. One of these criteria is the magnitude of relative error (MRE) which is defined for a project as follows:

$$MRE_i = \frac{|Actual\ Effort_i - Predicted\ Effort_i|}{Actual\ Effort_i} * 100\%$$

MRE is calculated for COCOMO II as well as for the proposed neuro-fuzzy approach. It is found that MRE obtained for the proposed model is quite less as compared to MRE obtained by COCOMO II.

Another broadly used and more accurate measure is the pred (l) which is defined as follows:

$$pred\ (l) = \frac{k}{N} * 100$$

Where, N is the total number of projects and k is the number of projects whose MRE is less than or equal to l.

7. Implementation and results

- This research will apply Constructive Cost Model (COCOMO-II) using Mamdani Fuzzy Inference System (FIS).

Table 1
Comparison between existing model and proposed model

Performance	Using 13GMFS	Using 11GMFS	Fuzzy	Cocomo	Fuzzy Cocomo-II	Neuro-Fuzzy	Neuro-Fuzzy With Cocomo-II
MMRE	26.90	30.38	33.21	37.92	41.98	46.25	49.38
PRED	53	36	24	15	12	10	7

- The results were analyzed by the criterion – MMRE (Mean Magnitude of Relative Error).
- Less MMRE means the result is more accurate.

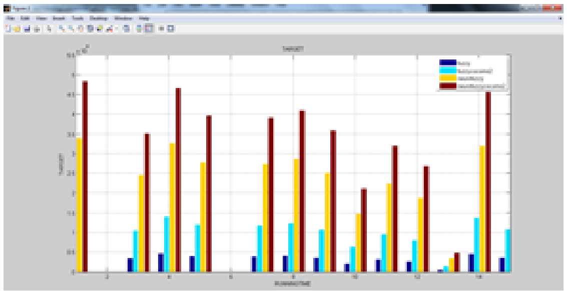

Fig. 4. Target

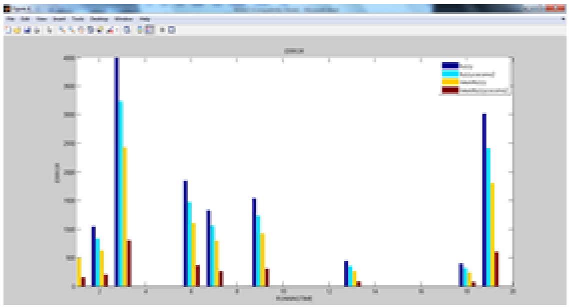

Fig. 5. Error rate

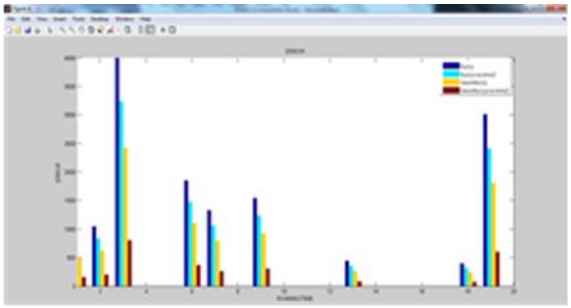

Fig. 6. MMRE

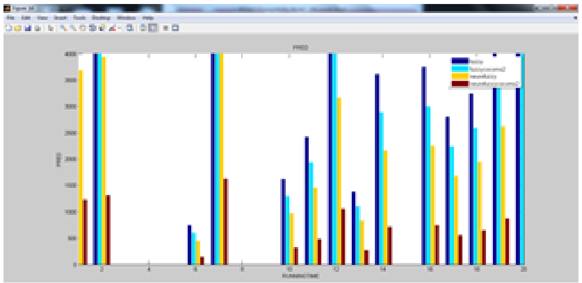

Fig. 7. PRED

8. Conclusion

A critical issue for project managers is the accurate and reliable estimates of the required software development effort, particularly in the early stages of the software development life cycle. Software effort drivers generally have properties of uncertainty and ambiguity when they are measured by human judgment. Cost drivers in algorithmic software cost estimation are frequently expressed through linguistic assessments and they generally represent high level concepts for which a single, exact measurement scale is not available. This motivates the use of neuro-fuzzy techniques to model evaluated inputs and their assessment procedures. To date, neuro-fuzzy logic modeling techniques have been shown to be the most effective approximation method to handle imprecise data. This work suggests a new approach for finding a software projects development effort. The major difference between our work and previous works is that neuro-fuzzy technique is used for software development effort estimation and then it's validated with collected data. Benefits of neural network and fuzzy logic are combined and learning ability and good generalization are obtained. The major benefit of this model is its good interpretability by using the fuzzy rules and another great benefit of this research is that it can put together expert knowledge project data and the learning ability of neural network model into one common framework that may have a wide range of applicability in software estimation. The results showed that neuro-fuzzy system is much superior to the previous methods.

References

[1] Idri, A. and Khoshgoftaar, T. M. and Abran, A., (2002), "Investigating Soft Computing in Case-based Reasoning for Software Cost Estimation", Engineering Intelligent Systems for Electrical Engineering and Communications, Vol. 10, No. 3, pp.147-157.

[2] Luiz Fernando Capretz and Venus Marza,Improving Effort Estimation by Voting Software Estimation Models Hindawi Publishing Corporation Advances in Software Engineering Volume 2009.

[3] Neha Sharma, Amit Sinhal, Bhupendra Verma, "Software Assessment Parameter Optimization using Genetic Algorithm", International Journal of Computer Applications, Vol. 72, No.7, pp. 8-13, May 2013.

[4] J. N. V. R Swarup Kumar, Aravind Mandala, M. Vishnu Chaitanya, G. V. S. N. R.V Prasad, "Fuzzy logic for Software Effort Estimation Using Polynomial Regression as Firing Interval", International Journal of Computer Technology Applications, Vol. 2, No. 6, pp. 1843-1847, Dec. 2011.

[5] Moataz A. Ahmed, Moshood Omolade Saliu, Jarallah AlGhamdi, Adaptive Fuzzy Logic-based framework for software effort prediction, Information and Software Technology 47(2005) 31-48.

[6] Cuauhtemoc Lopez-Martin, Cornelio Yanez-Marquez, Agustin Gutierrez-Tornes, Predictive accuracy comparison of Fuzzy models for software development effort of small programs, the Journals of systems and software 81(2008) 949-960.

[7] F.S. Gharehchopogh, "Neural networks application in software cost estimation: A case study", IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA), pp. 69-73, 15-18 June 2011.

[8] Wei Lin Du, Danny Ho and Luiz Fernando Capretz, "Improving Software Effort Estimation Using Neuro-Fuzzy Model with SEER-SEM", Global Journal of Computer Science and Technology, Vol. 10, No. 12, Pp. 52-64, Oct 2010.

[9] Pichai Jodpimai, Peraphon Sophatsathit and Chidchanok Lursinsap, "Estimating Software Effort with Minimum Features using Neural Functional Approximation", ICCSA.2010.

[10] X. Huang, Danny Ho, J. Ren, L.F. Capretz, "Improving the COCOMO model using a neuro-fuzzy approach",Applied Soft Computing , Vol.7, Issue 1, 2007, pp. 29-40.

[11] A.R. Gray, S.G. Mac Donell, A comparison of techniques for developing predictive models of software metrics, Inf. Software Technol. 39 (1997) 425–437.

[12] A. Idri, T.M. Khoshgoftaar, A. Abran, Can neural networks be easily interpreted in software cost estimation? In Proceedings of IEEE International Conference on Fuzzy Systems, 2002, 1162–1167.

[13] G. Wittig, G. Finnie, Estimating software development effort with connectionist models, Inf. Software Technol. 39 (1997) 469–476.

[14] Putnam L. H. and Myers W., Measures for Excellence, Prentice Hall, Englewood Cliffs, NJ, 1992.

[15] S. Mitra, Y.Hayashi, "Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework", IEEE Transactions on Neural Networks, Vol.11, No.3, 2000, pp. 748-768.

[16] D. Nauck, F. Klawonn, R. Kruse, "Foundations of Neuro-Fuzzy Systems", Chichester, 97.

[17] D. Nauck, "A Fuzzy Perceptron as a Generic Model for Neuro-Fuzzy Approaches", In Proceedings of Fuzzy-Systeme"94, 2nd GI-Workshop, Munich, Semen Corporation, 1994.

[18] M.O. Saliu, "Adaptive Fuzzy Logic Based Framework for Software Development Effort Prediction", A Thesis Presented to the DEANSHIP OF GRADUATE STUDIES, King Fahd University of Petroleum & Minerals Dhahran, April 2003.

[19] Chen Z., Menzies T., and Port D., "Feature Subset Selection Can Improve Software Cost Estimation Accuracy," in Proceedings of Workshop Predictor Models in Software Engineering, California, pp. 245-248, 2005.

[20] I. Attarzadeh, Siew Hock Ow, "A novel soft computing model to increase the accuracy of software development cost estimation", IEEE 2nd International Conference on Computer and Automation Engineering (ICCAE), Vol. 3, pp. 603 - 607, 26-28 Feb. 2010.