Automation of Software Cost Estimation using Neural Network Technique

Gaurav Kumar
Department of Computer Science & Engineering
Guru Jambheshwar University of Science & Technology
Hisar, Haryana, India

Pradeep Kumar Bhatia
Department of Computer Science & Engineering
Guru Jambheshwar University of Science & Technology
Hisar, Haryana, India

ABSTRACT

Software cost estimation is one of the most challenging tasks in software engineering. Over the past years the estimators have used parametric cost estimation models to establish software cost, however the challenges to accurate cost estimation keep evolving with the advancing technology. A detailed review of various cost estimation methods developed so far is presented in this paper. Planned effort and actual effort has been comparison in detail through applying on NASA projects. This paper uses Back-Propagation neural networks for software cost estimation. A model based on Neural Network has been proposed that takes KLOC of the project as input, uses COCOMO model parameters and gives effort as output. Artificial Neural Network represents a complex set of relationship between the effort and the cost drivers and is a potential tool for estimation. The proposed model automates the software cost estimation task and helps project manager to provide fast and realistic estimate for the project effort and development time that in turn gives software cost.

General Terms

Software Engineering, Software Planning.

Keywords

Back Propagation Neural Network, COCOMO Model, Software Cost Estimation.

1. INTRODUCTION

Software Cost is basically consists of:

- *Manpower* i.e. number of engineering and management personnel allocated to the project as a function of time.
- *Duration* i.e. the amount of time required to complete the project.
- *Effort* i.e. the engineering and management effort required to complete a project usually measured in person-months.

Software cost estimation usually fails to accurately predict the actual costs or the time needed to develop the system. Software cost estimation models have two problems. Estimating software development costs with accuracy is very difficult. The most common approach for improving software cost estimates is to use empirical models. It becomes more challenging to predict the costs and schedule at the beginning of the project. Early prediction of completion time is necessary for proper advance planning and to avoid any type of risk of a project.

Software cost estimation is the set of procedures and techniques with a set of inputs that an organization uses to achieve a software cost estimate in terms of effort, manpower, duration etc. Generally, inaccurate estimates are because of:

- Problem with requirements
- System Size
- Maintenance Issues
- Software Process and Process Maturity
- Project Monitoring and Control
- Lack of historical data
- Lack of application domain expertization

Here, we have considered the features of the cost estimation problem and have proposed a novel Neuro-COCOMO model. The major difference between our work and previous works is that we have combined the COCOMO model with neural network technique into one scheme and have validated our approach with NASA projects data.

2. BACKGROUND

2.1 COCOMO Model

The COCOMO model can be categorized into basic, intermediate, and detailed model. The basic model represents quick, early, and rough estimates of effort. The intermediate and detailed model includes more information in the form of cost drivers. The inputs of COCOMO model are: (1) the estimated size of the software product in thousands of Delivered Source Instructions (KDSI) adjusted for code reuse; (2) the project development mode given as a constant value B (also called the scaling factor); and (3) 15 cost drivers. The project development mode depends on one of the three categories of software development modes: organic, semi-detached, and embedded. Each rating has a corresponding real number (effort multiplier), based upon the factor and the degree to which the factor can influence productivity [23, 26, 29]. The estimated effort in person-months (PM) for the intermediate COCOMO is given as:

$$Effort = A \times [Size]^B \times \prod_{i=1}^{15} EM_i$$

Constant A is known as productivity coefficient. The effort multipliers corresponding to the cost drivers are incorporated into the effort estimation formula by multiplying them together.

Intermediate COCOMO model is the most widely used version. According to the researchers, intermediate COCOMO model has estimation accuracy that is greater than the basic version, and at the same time comparable to the detailed version. *Effort and Development Time can be calculated using following:*

$$E = a_i (KLOC)^{b_i} * EAF$$

$$D = c_i (E)^{d_i}$$

Where EAF is Effort Adjustment Factor. The product of all effort multipliers results in an *effort adjustment factor (EAF)*. Typical values for EAF range from 0.9 to 1.4. Coefficients used are shown in table 1. Values of the multipliers for the calculation of EAF are shown in table 2.

Table 1. Intermediate COCOMO coefficients

Project	**a_i**	**b_i**	**c_i**	**d_i**
Organic	3.2	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	2.8	1.2	2.5	0.32

Table 2. Values of the multipliers for calculation of Effort

Cost Drivers	**Ratings**					
	Very low	Low	Nominal	High	Very High	Extra High
Product Attributes						
Required S/w Reliability (RELY)	0.75	0.88	1	1.15	1.4	-
Size of Application Database (DATA)	-	0.94	1	1.08	1.16	-
Complexity of the Product (CPLX)	0.7	0.85	1	1.15	1.3	1.65
Computer Attributes						
Run Time Performance Constraints (TIME)	-	-	1	1.11	1.3	
Memory Constraints (STOR)	-	-	1	1.06	1.21	
Virtual Machine Volatility (VIRT)	-	0.87	1	1.15	1.3	-
Turnaround Time (TURN)	-	0.87	1	1.07	1.15	-
Personal Attributes						
Analyst Capability (ACAP)	1.46	1.19	1	0.86	0.71	-
Application Experience (AEXP)	1.29	1.13	1	0.91	0.82	-
Programmer Capability (PCAP)	1.42	1.17	1	0.86	0.7	-
Virtual M/c Experience (VEXP)	1.21	1.1	1	0.9	-	-
Programming Language Experience (LEXP)	1.14	1.07	1	0.95	-	-
Project Attributes						
Modern Programming Practices (MODP)	1.24	1.1	1	0.91	0.82	-
Use of Software Tools (TOOL)	1.24	1.1	1	0.91	0.83	-
Required Development Schedule (SCED)	1.23	1.08	1	1.04	1.1	-

2.2 Artificial Neural Network and MatLab

An Artificial Neural Network is composed of a large simple processor ‘units’ where each unit is having a small amount of local memory. These units are connected by unidirectional communication channels called ‘connections’, which carries numeric data. The units operate only on their local data and on the inputs they receive via the connections. In Multiple-layer networks, the number of layers determines the superscript on the weight matrices [27]. Two-layer tansig/purelin neural network as shown in figure 1 can approximate any function with a finite number of discontinuities when sufficient neurons in the hidden layer are given. MATLAB (MATrix LABoratory) is a numerical computing environment and fourth-generation programming language, which is developed by MathWorks [13]. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science.

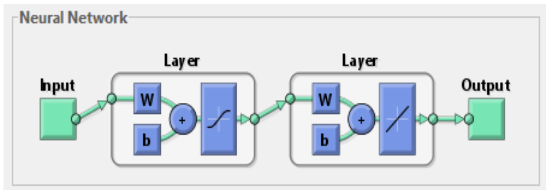

Fig 1. Multilayer feed forward neural network diagram

MATLAB is used as a tool for high-productivity research, development, and analysis. MATLAB features a family of add-on application-specific solutions called toolboxes that allows learning and applying specialized technology. Toolboxes are broad collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems.

3. LITERATURE REVIEW

Many cost estimation methods are available. Still research is going on in this area in order to find more accurate cost estimation method. In this section, the work done in this field is presented.

Neha Sharma et. al. (2013) proposed model to assess parameters for NASA software project dataset using a genetic algorithm (GA). The result shows that the three models (Organic model, Semi-detached model and Embedded model) takes much larger time and performs inferior than proposed model [1].

A. Idri and A. Zahi (2013) evaluate and compare the classical analogy and fuzzy analogy for software cost estimation on a web software dataset. They estimated accuracy, tolerance of imprecision and uncertainty of cost drivers showing the usefulness of Fuzzy Analogy for software cost estimation [2].

M. Azzeh (2013) analyzes the potential of Use Case Point estimation model for global projects and uses this as a basis to discuss three proposed factors (Global team trust, Global team composition and Culture value) that helps in managing the global software project development [3].

A. Kaushik et. al. (2013) mapped the COCOMO model by using a Feed Forward Back Propagation neural network. The proposed model takes identity function at the input layer and sigmoid function at the hidden and output layer. The model uses COCOMO and COCOMO NASA 2 dataset to train and test the network [4].

Anupama Kaushik et. al. (2012) made an analysis to use the fuzzy logic in the COCOMO model and provided in-depth review and comparison of software effort estimation models. They presented an overview of fuzzy approaches in COCOMO's effort estimation [5, 6].

Surendra Pal Singh et. al. (2012) analyzes neural network using Bayesian Regularization training algorithm and produces reduced condition numbers as compared to the fuzzy model having membership functions. It is concluded that the neural network model using Bayesian Regularization training algorithm is a more stable model than the fuzzy model having membership functions [7].

Abeer Hamdy (2012) developed an adaptive fuzzy model for software effort estimation. A fuzzy logic-based component with COCOMO81 intermediate model is used to improve its accuracy and sensitivity. The proposed model uses genetic algorithms (GA) to tune the fuzzy sets parameters [8].

Swarup Kumar et. al. (2011) proposed fuzzy software cost estimation model that handles ambiguousness, obscurity and a comparison is made with other popular software cost estimation models. Fuzzy logic method is used to address the problem of obscurity and vagueness exists in software effort drivers to estimate software effort [9].

Gharehchopogh (2011) made a case study for software cost estimation using Neural Network (NN) architecture for predicting necessary effort of new software. The results indicate that the NN model offers the very best algorithmic method to predict and estimate software costs [10].

Vishal Sharma and Harsh Kumar Verma (2010) presented an optimized fuzzy logic based framework for software development effort prediction. Fuzziness is incorporated into the measurements of size, mode of development for projects and the cost drivers contributing to the overall development effort [11].

Iman Attarzadeh et. al. (2010) proposed a model for handling imprecision and uncertainty by using the fuzzy logic systems. The proposed fuzzy logic model shows better software effort estimate evaluation criteria as compared to the traditional COCOMO. The results demonstrate that applying fuzzy logic method to the software effort estimation is a feasible approach to addressing the problem of uncertainty and vagueness [12].

Prasad Reddy (2010) presented a Particle Swarm Optimization Algorithm (PSOA) to fine tune the fuzzy estimate for the development of software projects. The efficiency of the developed models is tested on 10 NASA software projects, and COCOMO 81 database for assessing software cost estimation [14].

J. S. Pahariya et. al. (2009) presented computational intelligence techniques for software cost estimation using Genetic Programming (GP). Three linear ensembles based on (i) arithmetic mean (ii) geometric mean and (iii) harmonic mean was implemented. Proposed recurrent architecture for Genetic Programming provides best results as compared to other techniques considered [15].

A. S. Andreou et. al. (2008) addresses the issue of software cost estimation through fuzzy decision trees to acquire accurate and reliable effort estimation for project resource allocation and control. Approximately 1000 project data records are selected for analysis and experimentation, with fuzzy decision trees [16].

Ch. Satyananda Reddy et. al. (2008) shows the accuracy of the effort prediction using Radial Basis Function Neural Network (RBFN) i.e. used for functional approximation. The proposed network found that the RBFN designed with the K-means clustering algorithm performs better in terms of cost estimation accuracy [17].

Madhu S. Nair and Jaya vijayan (2008) proposed a simplified neural model to optimize a project team to attain maximum throughput as well as to obtain high quality software. Artificial Neural Networks is used to train the software professionals and make them perform at high level of standards [18].

Mitat Uysal (2008) shows that Simulated Annealing algorithm can be used to estimate the optimal parameters of the effort components of software projects. If a larger search space is built, it would take more time for computations; and convergence of search may become very slow. Conversely, if the search space is set too small, the optimal parameters probably could not been found [19].

Harish Mittal and Pradeep Bhatia (2007) estimated size in person hours by using Fuzzy logic to find fuzzy functional points and then the result is defuzzified to get the functional points. Triangular fuzzy numbers was used to represent the linguistic terms in Function Point Analysis (FPA) complexity matrices. The developed models were tested on NASA software projects [20, 22].

Xishi Huang et. al. (2003, 2007) proposed a novel Neuro-Fuzzy Constructive Cost Model (COCOMO) for software estimation. It allows inputs to be continuous-rating values and linguistic values, therefore avoiding the problem of similar projects having different estimated costs. Using industry project data, validation shows that the model greatly improves the estimation accuracy as compared to COCOMO model [21, 30].

Alaa F. Sheta (2006) estimated the effort required for the development of software projects using Genetic Algorithms (GAs). The proposed model structure was tested on NASA software project dataset. A modified version of the COCOMO model is provided to explore the effort computation [24].

Moataz A. Ahmed et. al. (2005) predicted software effort using an adaptive fuzzy logic framework. In the framework, the training and adaptation algorithms tolerates imprecision, explains prediction rationale through rules, incorporates experts knowledge, offers transparency in the prediction system, and could adapt to new environments as data for the new environment becomes available [25].

A. Idri et. al. (2004) presented the cost estimation models based on artificial neural networks. The COCOMO'81 dataset was used to train and test the Radial Basis Function network (RBFN). The fuzzy rules express the information encoded in the architecture of the network [28].

S. Vijayakumar (1997) presented justification of the requirement for a database as an aid to resource estimations for Ministry of Defence Procurement Executive software intensive projects. The research was initiated to identify and analyse the variables which influence the activities that constitute software development and determine the cost of software [31].

Xiangzhu Gao and B. Lo (1996) examines the special characteristics of Computer Assisted Learning (CAL) systems and analyzes the additional considerations needed if Function point analysis (FPA) is used to size CAL systems for effort estimation. Real world data was used to test the model [32].

J. E. Matson et. al. (1994) presented an assessment of several published statistical regression models that relate software development effort to software size measured in function points. The research describes appropriate statistical procedures in the context of a case study based on function point data for 104 software development projects and discusses limitations of the resulting model in estimating development effort [33].

A. R. Venkatachalam (1993) used an artificial neural network approach to model the software cost estimation expertise and results was compared with the COCOMO model [34].

4. RESEARCH METHODOLOGY

Through a detailed literature review, the parameters that affect the cost of a project, the methods developed earlier for estimation, their good practices and bad practices has been discussed in detail. COCOMO which is the most popular tool for estimating software cost and uses lines of code to assess software size has been discussed. A COCOMO intermediate model is implemented with multiple projects for the said problem. This model is rectified in the form of Neural Network model and then a detailed comparison between the planned effort and actual effort has been made.

Data for analysing and implementing cost estimation model is derived from NASA projects. From where, input data is obtained in the form of various parameter values of COCOMO-Intermediate model with actual effort derived from multiple projects. A Neural Network is made to set the values of parameters with planned effort and actual effort. An artificial neural network is modelled as a massively parallel-interconnected network of elementary processors or neurons. It has been shown that a three-layer feed forward network can generate arbitrary complex decision regions. The multi-layered neural networks operate in two modes: Training and testing. In the training mode, a set of training data is used to adjust the weights of the network interconnections so that the network responds in a specified manner. In the testing mode, the trained network is evaluated by the test data which comprise of data from NASA projects. Evaluation criteria of software effort estimation include [4, 5, 6]:

$$Variance\ Accounted\ For\ (VAF)\% = \left(1 - \frac{var(E - \hat{E})}{var\ E}\right) \times 100$$

$$Balance\ Relative\ Error\ (BRE) = \frac{|E - \hat{E}|}{\min\ (E, \hat{E})}$$

$$Magnitude\ of\ Relative\ Error\ (MRE)\% = \frac{|E - \hat{E}|}{E} \times 100$$

$$Mean\ Magnitude\ of\ Relative\ Error\ (MMRE) = \frac{1}{N}\sum_i \frac{E_i - \hat{E}_i}{E_i}$$

Where E, $\hat{E}$ is Estimated Effort and Actual Effort respectively.

A model provides better estimation which is having higher VAF, lower BRE, lower MRE and lower MMRE.

Input data (60 nos) is taken from the NASA projects having COCOMO attribute values with KLOC and actual Effort. It is in the form of:

- Nominal,High,Very_High,Nominal,Nominal,Low,Nominal ,High,Nominal,Very_High,Low,Nominal,High,Nominal,Lo w,70,278
- Very_High,High,High,Very_High,Very_High,Nominal,No minal,Very_High,Very_High,Very_High,Nominal,High,Hi gh,High,Low,227,1181

Back Propagation Neural Network is created using the following function:

```
net=newff(input',out_data',16,{},'trainlm');
```

The input data is fed into Neural Network for training and is implemented using the values as shown in table 3. For testing Neural Network, test data is taken from the NASA projects.

Table 3. Experimental values taken for implementation

Parameter	Values
Convergence Objective	0.01
Learning rate	0.005
Training method used	trainlm (Levenberg-Marquardt)
No. of training data	60
No. of testing data	27
No. of epoch taken to converge	2000
Gradient	1.04

5. RESULT ANALYSIS

Proposed model provides the estimated effort, development time, balance relative error, magnitude of relative error w.r.t. KLOC and actual effort as shown in table 4. A graph showing planned effort, actual effort and development time is shown in figure 2 where x axis represents KLOC and y axis represents Effort with Time.

Table 4. Estimated Effort, Development Time, evaluation parameters on NASA Projects

S. No.	KLOC	Actual Effort	Estimated Effort	Dev. Time	BRE	MRE %
1	70	278	278	17.9197	0.0002	0.0168
2	227	1181	1236.7	30.2153	0.0471	4.5005
3	177.9	1248	1202.7	29.9226	0.0376	3.7634
4	115.8	480	539.8	22.6058	0.1246	11.0767
5	29.5	120	98.2	14.2883	0.2217	22.1708
6	19.7	60	64.3	12.1623	0.0714	6.6615
7	66.6	300	290.5	18.199	0.0327	3.2662

8	5.5	18	16.8	7.3102	0.069	6.9033
9	10.4	50	32.9	9.4258	0.5212	52.1193
10	14	60	44.9	10.6127	0.336	33.6029
11	16	114	75.1	12.9012	0.5185	51.8476
12	6.5	42	30.2	9.1281	0.3904	39.0416
13	13	60	54.4	11.4137	0.1032	10.3222
14	8	42	32.7	9.4036	0.2858	28.5751
15	90	450	360.5	19.6267	0.2484	24.8352
16	15	90	54.9	11.4549	0.6392	63.9215
17	38	210	151	16.8231	0.3911	39.1109
18	10	48	28.1	8.8806	0.7082	70.825
19	161.1	815	862.2	26.6315	0.0579	5.4699
20	48.5	239	182.7	18.0878	0.3083	30.8277
21	32.6	170	120.4	15.4365	0.4122	41.2216
22	12.8	62	45.1	10.6304	0.3745	37.4523
23	15.4	70	54.8	11.4445	0.278	27.8003
24	16.3	82	58.1	11.7068	0.4104	41.0417
25	35.5	192	131.6	15.9704	0.4585	45.8452
26	25.9	117.6	85.7	13.5653	0.3726	37.2593
27	24.6	117.6	81.2	13.2894	0.4489	44.8854
28	7.7	31.2	24	8.3605	0.3015	30.1483
29	9.7	25.2	30.5	9.1674	0.2123	17.5123
30	2.2	8.4	6.4	5.0716	0.3057	30.5674
31	3.5	10.8	10.5	6.1039	0.031	3.0983
32	8.2	36	25.6	8.573	0.4057	40.5715
33	66.6	352.8	290.5	18.199	0.2144	21.441
34	150	324	491.4	21.8753	0.5168	34.0706
35	100	360	418.2	20.6741	0.1617	13.9177
36	100	215	475.9	21.6304	1.2134	54.8208
37	100	360	200	15.971	0.7996	79.9629
38	15	48	25.9	8.6109	0.8527	85.2665
39	32.5	60	117.5	15.2931	0.9576	48.9176
40	31.5	60	62.7	12.0448	0.0444	4.2477
41	6	24	9.9	5.9741	1.4244	142.4398
42	11.3	36	25.2	8.521	0.4284	42.8433
43	20	72	28.5	8.9316	1.5241	152.4079
44	20	48	35	9.6583	0.3697	36.9655
45	7.5	72	42.1	10.3571	0.7095	70.9481
46	302	2400	2419.3	30.2497	0.008	0.7978
47	370	3240	4068.6	35.7241	0.2557	20.3648
48	219	2120	1509.6	32.3997	0.4044	40.4357
49	50	370	234.2	19.8771	0.5801	58.0133
50	101	750	602.7	23.495	0.2444	24.4395
51	190	420	436.9	20.9935	0.0403	3.8749
52	47.5	252	158.1	17.1225	0.5936	59.3604
53	21	107	146.6	16.6383	0.3704	27.0293
54	423	2300	1731.7	27.18	0.3282	32.8189
55	79	400	279.8	17.9618	0.4295	42.9463
56	284.7	973	1353.8	31.1874	0.3913	28.126
57	282.1	1368	1139.6	29.3628	0.2005	20.0467
58	78	571.4	548.7	22.7349	0.0415	4.1463
59	11.4	98.8	57.3	11.64	0.7252	72.5153
60	19.3	155	99.5	14.361	0.5571	55.7103

Value of VAF %=93.5542, MMRE=0.3642

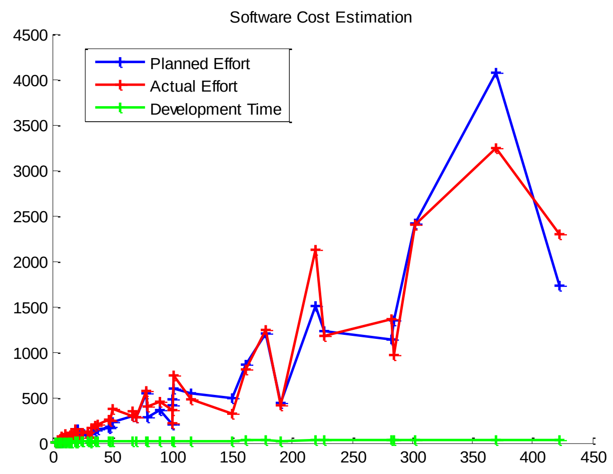

Figure 2. Graph showing Planned Effort vs. Actual Effort with Development Time.

6. CONCLUSION & FUTURE SCOPE

In this paper, we have analyzed and studied the software cost estimation model using COCOMO model based on calculating Planned Effort and Development time. The output values are calculated through various attribute values lying between very low to very high. Hereby, in the research work we have done a detailed literature review of the cost estimation models developed earlier. A detailed comparison between planned effort and actual effort has been shown through a graph by applying input values obtained from NASA projects. The work done shows how much planned effort differs from the actual effort on the basis of realistic values obtained. We have proposed a model that uses a Neural Network which takes input values obtained through COCOMO model and gives estimated effort and development time. The proposed model may benefit the project manager to provide more realistic estimate for the project effort and development time that implies software cost. Hence, a software cost estimation model has been proposed to give cost using neural network technique.

Although proposed model can be used for all type of COCOMO modes, still there is a scope of implementing the model for all type of modes. For this implementation, some valid data is required like the data used in our proposed model. In the future, we are going to extend the model for other type of modes and optimize the cost estimation using Genetic Algorithms and Fuzzy techniques.

7. REFERENCES

[1] 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.

[2] A. Idri, A. Zahi, “Software cost estimation by classical and Fuzzy Analogy for Web Hypermedia Applications: A replicated study”, IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 207-213, 16-19 April 2013.

[3] M. Azzeh, “Software cost estimation based on use case points for global software development”, 5th IEEE International Conference on Computer Science and Information Technology (CSIT), pp. 214-218, 27-28 March 2013.

[4] A. Kaushik, A. K. Soni, Rachna Soni, “A Simple Neural Network Approach to Software Cost Estimation”, Global Journals of Computer Science & Technology, Vol. 13, Issue 1, Version 1, pp. 23-30, 2013.

[5] Anupama Kaushik, A. K. Soni, Rachna Soni, “A Comparative Study on Fuzzy Approaches for COCOMO’s Effort Estimation”, International Journal of Computer Theory and Engineering, Vol. 4, No. 6, pp. 990-993, Dec. 2012.

[6] A. Kaushik, A. K. Soni, R. Soni, "An adaptive learning approach to software cost estimation", National Conference on Computing and Communication Systems (NCCCS), pp. 1-6, 21-22 Nov. 2012

[7] Surendra Pal Singh, Prashant Johri, “A Review of Estimating Development Time and Efforts of Software Projects by Using Neural Network and Fuzzy”, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, Issue 10, pp. 306-310, Oct. 2012.

[8] Abeer Hamdy, “Fuzzy Logic for Enhancing the Sensitivity of COCOMO Cost Model”, Journal of Emerging Trends in Computing and Information Sciences, Vol. 3, No. 9, pp. 1292-1297, Sep. 2012.

[9] 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.

[10] 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.

[11] Vishal Sharma, Harsh Kumar Verma, “Optimized Fuzzy Logic Based Framework for Effort Estimation in Software Development”, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 2, No 2, pp. 30-38, March 2010.

[12] 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.

[13] MatLab R2010 Neural Network Tool Box Product Help

[14] P. V. G. D Prasad Reddy, “Particle Swarm Optimization in the fine-tuning of Fuzzy Software Cost Estimation Models”, International Journal of Software Engineering (IJSE), Vol. 1, Issue 2, pp. 12-23, 2010.

[15] J. S. Pahariya, V. Ravi, M. Carr, “Software Cost Estimation using Computational Intelligence Techniques”, IEEE World Congress on Nature & Biologically Inspired Computing (NaBIC), pp. 849-854, 9-11 Dec, 2009.

[16] A. S. Andreou, E. Papatheocharous, “Software Cost Estimation using Fuzzy Decision Trees”, 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 371-374, 15-19 Sept. 2008.

[17] Ch. Satyananda Reddy, P. Sankara Rao, KVSVN Raju, V. Valli Kumari, “A New Approach For Estimating Software Effort Using RBFN Network”, International Journal of Computer Science and Network Security, Vol. 8, No.7, pp. 237-241, July 2008.

[18] Madhu S. Nair, Jaya Vijayan, “Simplified Neural Model for the Software Development Team Optimization”, International Arab Journal of Information Technology, Vol. 5, No. 2, April 2008.

[19] Mitat Uysal, “Estimation of the Effort Component of the Software Projects Using Simulated Annealing Algorithm”, World Academy of Science, Engineering and Technology, Vol. 17, pp. 234-237, 2008.

[20] Harish Mittal, Pradeep Bhatia, “Optimization Criteria for Effort Estimation using Fuzzy Technique”, CLEI Electronic Journal , Vol. 10, No. 1, Paper 2, pp. 1-11, June 2007.

[21] Xishi Huang, Danny Ho, Jing Ren, Luiz F. Capretz, “Improving the COCOMO model using a neuro-fuzzy approach”, Elsevier Journal Applied Soft Computing, Vol. 7, pp. 29–40, 2007.

[22] Harish Mittal, Pradeep Bhatia, “A comparative study of conventional effort estimation and fuzzy effort estimation based on Triangular Fuzzy Numbers”, International Journal of Computer Science and Security, Volume 1, Issue 4, pp. 36-47, 2007.

[23] K. K. Aggarwal, Yogesh Singh, Software Engineering, 3rd edition, New Age International Publishers, 2007.

[24] Alaa F. Sheta, “Estimation of the COCOMO Model Parameters Using Genetic Algorithms for NASA Software Projects”, Journal of Computer Science, Vol. 2, No. 2, pp. 118-123, 2006.

[25] Moataz A. Ahmed, Moshood Omolade Saliu, Jarallah AlGhamdi, “Adaptive fuzzy logic-based framework for software development effort prediction”, Elsevier Journal of Information and Software Technology, Vol. 47, pp. 31–48, 2005.

[26] Roger S. Pressman, “Software Engineering, A Practitioner’s Approach” Sixth Edition, McGraw-Hill, 2005.

[27] Simon Haykin, “Neural Networks: A Comprehensive Foundation”, Book by Pearson Education, Inc., 2004.

[28] A. Idri, S. Mbarki, A. Abran, “Validating and understanding software cost estimation models based on neural networks”, Proc. of IEEE International Conference on Information and Communication Technologies: From Theory to Applications, pp. 433-434, 19-23 April 2004.

[29] Ian Sommerville, "Software Engineering", Addison Wesley, 7th edition, 2004.

[30] Xishi Huang, L.F. Capretz, Jing Ren, D. Ho, “A neuro-fuzzy model for software cost estimation”, Proc. of IEEE 3rd International Conference on Quality Software, pp. 126 - 133, 6-7 Nov. 2003.

[31] S. Vijayakumar, “Use of historical data In software cost estimation”, IEEE Computing & Control Engineering Journal, Vol. 8, Issue 3, pp. 113-119, June 1997.

[32] Xiangzhu Gao, B. Lo, “A modified function point method for CAL systems with respect to software cost estimation”, Proc. of IEEE International Conference Software Engineering: Education and Practice, pp. 212 - 219, 24-27 Jan 1996.

[33] J. E. Matson, B. E. Barrett, J. M. Mellichamp, “Software development cost estimation using function points”, IEEE Transactions on Software Engineering, Vol. 20, Issue 4, pp. 275-287, Apr 1994.

[34] A.R. Venkatachalam, “Software cost estimation using artificial neural networks”, Proc. of International Joint Conference on Neural Networks (IJCNN), Vol. 1, pp. 987-990, 25-29 Oct. 1993.