ISSN : 2231-0711

Prediction of Software Development Effort Using RBNN and GRNN

Prasad Reddy P.V.G.D [1], Sudha K. R [2], Rama Sree P [3]

[1] *Dept. of CSSE, Andhra University, Visakhapatnam, 530003, Andhra Pradesh, INDIA*

[2] *Dept. of EE, Andhra University, Visakhapatnam, 530003, Andhra Pradesh, INDIA*

[3] *Dept. of CSE, Aditya Engineering College, JNTUK, Kakinada, 533003, Andhra Pradesh, INDIA.*

***Abstract*— Software development effort prediction is one of the most key activities in software industry. Many models have been proposed to build a relationship between software size and effort; however we still have problems for effort prediction. This is because project data, available in the primary stages of project is often inadequate, unpredictable, uncertain and unclear. The need for accurate effort estimation in software industry is an ongoing challenge. Artificial Neural Network models are more apt in such situations. The present paper is concerned with developing software effort prediction models based on artificial neural networks. The models are designed to improve the performance of the network that suits to the COCOMO Model. Artificial Neural Network models are created using Radial Basis and Generalized Regression. A case study based on the NASA 93 database compares the proposed neural network models with the Intermediate COCOMO. The results were analyzed using different criterions VAF, MMRE, MARE, VARE, Mean BRE and Prediction. It is observed that the Radial Basis Neural Network provided better results.**

***Keywords*— Intermediate COCOMO, Cost Estimation, Radial Basis Neural Networks, Generalized Regression Neural Networks.**

I. INTRODUCTION

In algorithmic cost estimation [14], costs and efforts are predicted using mathematical formulae. The formulae are resultant of some historical data [2]. The algorithmic cost model best known called COCOMO (COnstructive COst MOdel) was proposed by Barry Boehm in 1981[3]. It was developed from the analysis of sixty three (63) software projects. Boehm proposed three levels of the model called Basic COCOMO, Intermediate COCOMO and Detailed COCOMO [3], [15]. In the present paper we mainly focus on the Intermediate COCOMO.

A. *Intermediate COCOMO*

The Basic COCOMO model [3] is based on the relationship: Development Effort, DE = a*(SIZE) b where, SIZE is measured in thousand delivered source instructions. The constants a, b are dependent upon the 'mode' of development of projects. DE is measured in man-months. Boehm proposed 3 modes of projects [5]:

1) Organic mode: simple projects that engage small teams working in known and stable environments.

2) Semi-detached mode: projects that engage teams with a mixture of experience. It is in between organic and embedded modes.

3) Embedded mode: complex projects that are developed under tight constraints with changing requirements.

The accuracy of Basic COCOMO is limited because it does not consider the factors like hardware, personnel, use of modern tools and other attributes that affect the project cost. Further, Boehm proposed the Intermediate COCOMO [3], that adds accuracy to the Basic COCOMO by multiplying 'Cost Drivers' into the equation with a new variable: EAF (Effort Adjustment Factor) shown in Table 1.

TABLE I
DE FOR THE INTERMEDIATE COCOMO

Development Mode	Intermediate Effort Equation
Organic	$DE = EAF * 3.2 * (SIZE)^{1.05}$
Semi-detached	$DE = EAF * 3.0 * (SIZE)^{1.12}$
Embedded	$DE = EAF * 2.8 * (SIZE)^{1.2}$

The EAF term is the product of 15 Cost Drivers [15] that are listed in Table 2 .The multipliers of the cost drivers are Very Low, Low, Nominal, High, Very High and Extra High. For example, for a project, if RELY is Low, DATA is High , CPLX is extra high, TIME is Very High, STOR is High and rest parameters are Nominal then EAF = 0.75 * 1.08 *1.65*1.30*1.06 *1.0. If the category values of all the 15 cost drivers are "Nominal", then EAF is equal to 1.

TABLE III
INTERMEDIATE COCOMO COST DRIVERS WITH MULTIPLIERS

S. No	Cost Driver Symbol	Very low	Low	Nominal	High	Very high	Extra high
1	RELY	0.75	0.88	1.00	1.15	1.40	—
2	DATA	—	0.94	1.00	1.08	1.16	—
3	CPLX	0.70	0.85	1.00	1.15	1.30	1.65
4	TIME	—	—	1.00	1.11	1.30	1.66
5	STOR	—	—	1.00	1.06	1.21	1.56
6	VIRT	—	0.87	1.00	1.15	1.30	—
7	TURN	—	0.87	1.00	1.07	1.15	—
8	ACAP	—	0.87	1.00	1.07	1.15	—
9	AEXP	1.29	1.13	1.00	0.91	0.82	—
10	PCAP	1.42	1.17	1.00	0.86	0.70	—
11	VEXP	1.21	1.10	1.00	0.90	—	—
12	LEXP	1.14	1.07	1.00	0.95	—	—
13	MODP	1.24	1.10	1.00	0.91	0.82	—
14	TOOL	1.24	1.10	1.00	0.91	0.83	—
15	SCED	1.23	1.08	1.00	1.04	1.10	—

The 15 cost drivers are broadly classified into 4 categories [3], [15].

1) Product: RELY - Required software reliability
DATA - Data base size
CPLX - Product complexity

2) Platform: TIME - Execution time
STOR—main storage constraint
VIRT—virtual machine volatility
TURN—computer turnaround time

3) Personnel: ACAP—analyst capability
AEXP—applications experience
PCAP—programmer capability
VEXP—virtual machine experience
LEXP—language experience

4) Project: MODP—modern programming
TOOL—use of software tools
SCED—required development schedule

Depending on the projects, multipliers of the cost drivers will vary and thereby the EAF may be greater than or less than 1, thus affecting the Effort [12], [13], [15].

II. PROPOSED NEURAL NETWORK MODELS

A neural network [9] is a massive parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects [8], [4]:

1) Knowledge is acquired by the network from its environment through a learning process [10].

2) Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

In this section we are going to present the two network models [1] used for the case study i.e. Radial Basis Neural Network (RBNN) and Generalized Regression Neural Network (GRNN).

A. Radial Basis Neural Network

Radial Basis Neural Network (RBNN) consists of two layers: a hidden radial basis layer of S1 neurons, and an output linear layer of S2 neurons [1]. A Radial Basis neuron model with R inputs is shown in Fig. 1. Radial Basis Neuron uses the radbas transfer function. The net input to the radbas transfer function is the vector distance between its weight vector w and the input vector p, multiplied by the bias b. (The || dist || box in this figure accepts the input vector p and the single row input weight matrix, and produces the dot product of the two.)The transfer function for a radial basis neuron is given in "Eq. (1)".

$$radbas \quad (n) = e^{-n^2} \qquad (1)$$

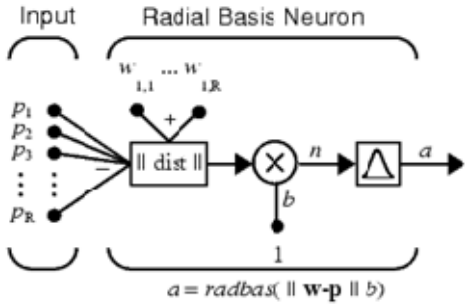

Fig. 1. Radial Basis neuron model

A plot of the radbas transfer function is shown in Fig. 2.

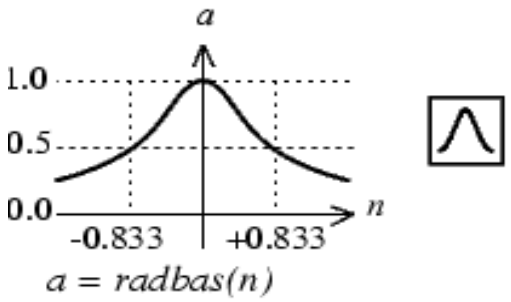

Fig. 2. radbas ( ) transfer function

The radial basis function has a maximum of 1 when its input is 0. As the distance between w and p decreases, the output increases. Thus, a radial basis neuron acts as a detector that produces 1 whenever the input p is identical to its weight vector w.The bias b allows the sensitivity of the radbas neuron to be adjusted. RBNN architecture is shown in Fig. 3.

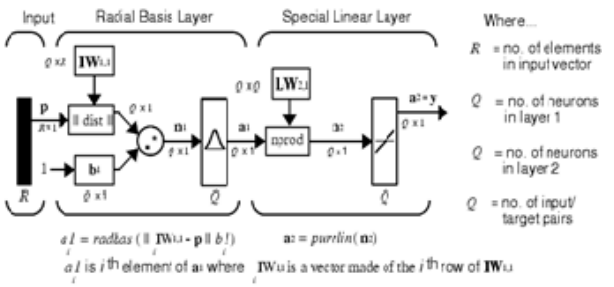

Fig. 3. Radial Basis Neural Network Architecture

The output of the first layer of this network can be obtained using "Eq.(2)".

$$a\{1\} = radbas(netprod(dist(net.IW\{1,1\}, p, net.b\{1\})) \quad (2)$$

If you present an input vector to this network, each neuron in the radial basis layer will output a value according to how close the input vector is to each neuron's weight vector. If a neuron has an output of 1, its output weights in the second layer pass their values to the linear neurons in the second layer. The second-layer weights LW 2,1 (or in code, LW{2,1}) and biases b2 (or in code, b{2}) are found by simulating the first-layer outputs a1 (A{1}), and then solving the linear expression "Eq.(3)".

$$[w\{2,1\}b\{2\}]*[A\{1\}; ones] = T \quad (3)$$

We know the inputs to the second layer (A{1}) and the target (T), and the layer is linear. We can use "Eq.(4)" to calculate the weights and biases of the second layer to minimize the sum-squared error.3

$$Wb = T/[P; ones(1, Q)] \quad (4)$$

Here Wb contains both weights and biases, with the biases in the last column. There is another factor called SPREAD used in the network. The user chooses SPREAD that is the distance an input vector must be from a neuron's weight. A larger SPREAD leads to a large area around the input vector where layer 1 neurons will respond with significant outputs. Therefore if SPREAD is small, the radial basis function is very steep, so that the neuron with the weight vector closest to the input will have a much larger output than other neurons. The network tends to respond with the target vector associated with the nearest design input vector.

B. Generalized Regression Neural Networks

A generalized regression neural network (GRNN) is often used for function approximation [11]. It has a radial basis layer and special linear layer. The architecture for the GRNN is shown in Fig. 4. It is similar to the radial basis network, but has a slightly different second layer.

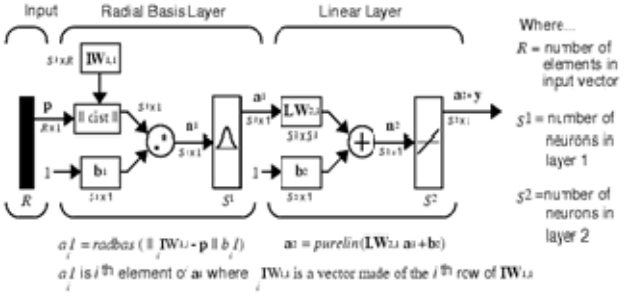

Fig. 4. Generalized Regression Neural Network Architecture

The first layer is just like that for Radial Basis networks. The second layer also has as many neurons as input/target vectors, but here LW{2,1} is set to T. Here the nprod box shown above produces S2 elements in vector n2. Each element is the dot product of a row of LW2,1 and the input vector a1. The user chooses SPREAD, the distance an input vector must be from a neuron's weight.

C. Advantages of Radial Basis Networks

1) Radial basis networks can be designed in a fraction of the time that it takes to train standard feed forward networks. They work best when many training vectors are available.
2) Radial Basis Networks are created with zero error on training vectors.

III. Various Criterions for Assessment of Estimation Models

1. Mean Absolute Relative Error (MARE)

$$MARE \quad (\%) \quad = \frac{\sum f(R_E)}{\sum f} \times 100 \quad (5)$$

2. Variance Absolute Relative Error (VARE)

$$VARE(\%) = \frac{\sum f(R_E - meanR_E)}{\sum f} \times 100 \quad (6)$$

3. Prediction (n)

Prediction at level n is defined as the % of projects that have absolute relative error less than n.

4. Balance Relative Error (BRE)

$$BRE = \frac{\left|E - \hat{E}\right|}{\min(E, \hat{E})} \quad (7)$$

5. Mean Magnitude of Relative Error (MMRE)

$$MMRE(\%) = \frac{1}{N}\sum_{i=1}^{N} MRE_i \times 100 \quad (8)$$

$$\text{Where} \qquad MRE = \left|\frac{\hat{E} - E}{\hat{E}}\right| \quad (9)$$

N = No. of Projects, E = estimated effort, $\hat{E}$ = actual effort

$$\text{Absolute Relative Error (RE)} = \left| \frac{\hat{E} - E}{\hat{E}} \right| \quad (10)$$

6) Variance Accounted For (VAF)

$$VAF\ (\%) = \left(1 - \frac{\text{var}\left(E - \hat{E}\right)}{\text{var}\ E} \right) \times 100 \quad (11)$$

A model which gives lower MARE "Eq. (5)" [12], [13] is better than that which gives higher MARE. A model which gives lower VARE is better than that which gives higher VARE "Eq. (6)". A model which gives lower BRE "Eq. (7)" is better than that which gives higher BRE. A model which gives higher Pred (n) is better than that which gives lower Pred (n). A model which gives lower MMRE "Eq. (8)" [12], [13] is better than that which gives higher MMRE. A model which gives higher VAF "Eq. (11)" [6] is better.

IV. Experimental Study

In carrying out our experiments, we have chosen the NASA 93 dataset taken from different NASA Centre's. Out of 93 projects, randomly selected 83 projects are used as training data. A Radial Basis Network and Generalized Regression Network are created .The two networks are tested using the entire 93 dataset. For creating radial basis network, newrbe ( ) is used and for creating generalized regression network, newgrnn( ) is used. The estimated efforts using Intermediate COCOMO, RBNN and GRNN are shown for some sample projects in Table 3. The Effort is calculated in man-months. Table 4 and Fig.5, Fig.6, Fig.7, Fig.8, Fig.9, Fig.10 & Fig.11 shows the comparisons of various models [7] basing on different criterions.

TABLE IIIII
Estimated Effort in Man Months of Various Models

Project ID	Actual Effort	Estimated Effort using		
		COCOMO	RBNN	GRNN
5	8.40	8.40	9.96	6.37
6	10.80	10.80	14.92	10.72
13	36.00	27.86	42.25	27.82
18	98.80	98.80	78.23	63.66
21	60.00	60.00	60.59	50.64
22	48.00	48.00	65.95	29.36
23	90.00	90.00	72.96	62.22
29	72.00	72.00	100.25	32.98
45	352.80	326.40	326.42	290.51
49	400.00	400.00	404.10	279.83
56	750.00	703.06	738.81	602.70
57	600.00	600.00	600.00	436.04
65	1248.00	1248.00	1248.00	1509.59
66	420.00	420.00	420.00	1236.66
67	2120.00	2120.00	2120.00	1139.56
79	480.00	480.00	465.14	278.48
82	599.00	599.00	596.62	387.30
84	480.00	480.00	1174.01	471.71
91	4178.20	4178.20	3375.24	2393.96

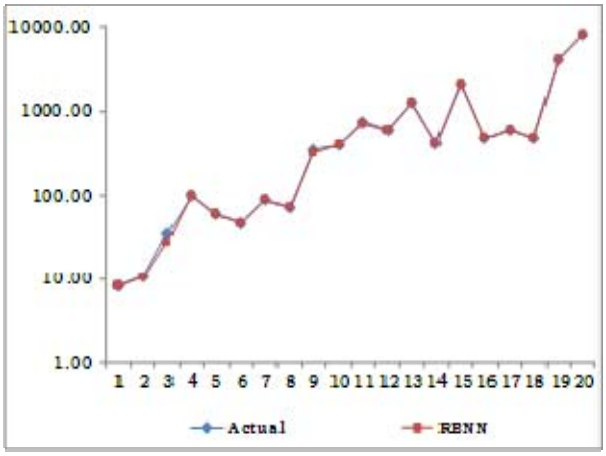

Fig. 5. Actual Effort versus RBNN Effort

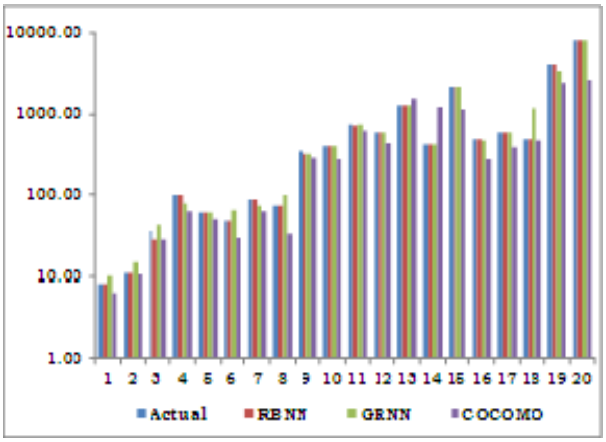

Fig. 6. Estimated Effort of various models versus Actual Effort

TABLE IV
Comparison of Various Models

Model	VAF	MARE	VARE	Mean BRE	MMRE	Pred(30) %
RBNN	98.477	7.26	5.18	0.293	28.5399	91
GRNN	98.476	18.60	11.24	0.339	30.8303	80
COCOMO	33.645	47.22	46.89	0.776	59.5027	53

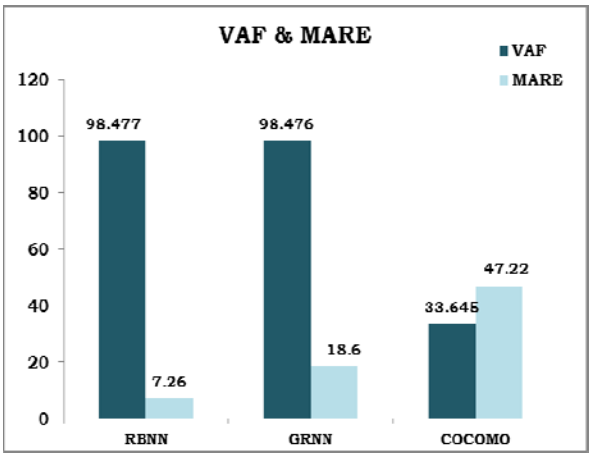

Fig. 7. Comparison of VAF & MARE against various models

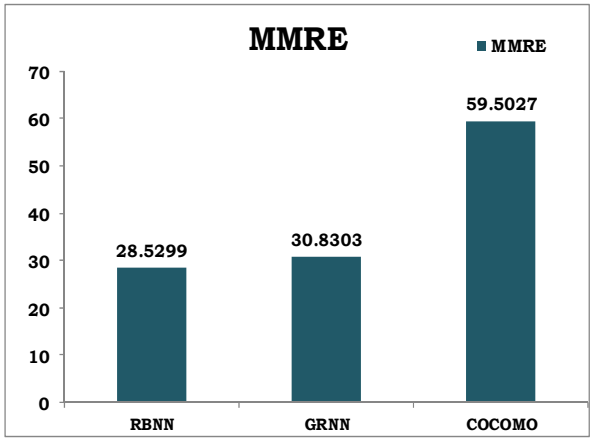

Fig. 10. Comparison of MMRE against various models

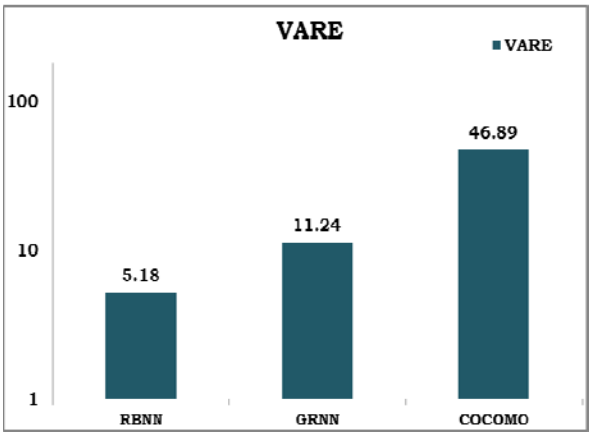

Fig. 8. Comparison of VARE against various models

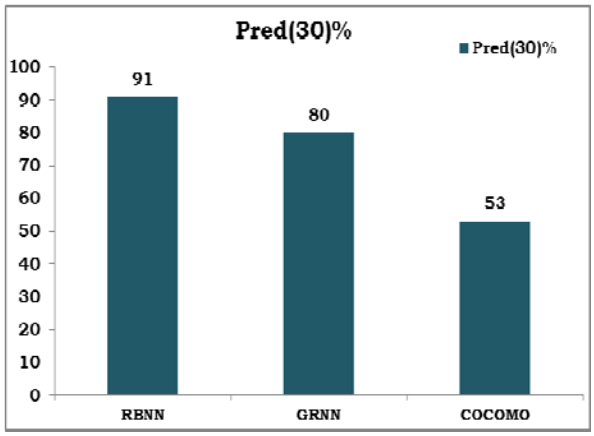

Fig. 11. Comparison of Pred(30)% against various models

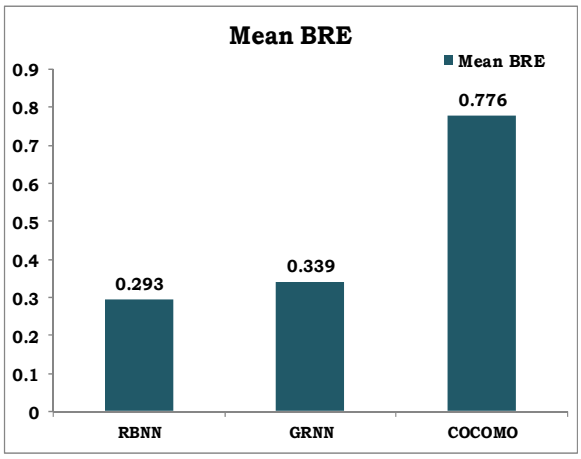

Fig. 9. Comparison of Mean BRE against various models

V. Conclusions

Referring to Table 4, we see that Radial Basis Neural Networks yields better results for maximum criterions when compared with the other models. Thus, basing on VAF, MARE, VARE, Mean BRE, MMRE & Pred(30) we come to a conclusion that RBNN is better than GRNN or Intermediate COCOMO. Therefore we proved that it's better to create a Radial Basis Neural Network for software effort prediction using some training data and use it for effort estimation for all the other projects.

References

[1] Ali Idri, Alain Abran, Samir Mbarki, "Validating and Understanding Software Cost Estimation Models based on Neural Networks", 0-7803-8482-2/04,2004 IEEE

[2] Angelis L, Stamelos I, Morisio M, Building a software cost estimation model based on categorical data, Software Metrics Symposium, 2001-Seventh International Volume (2001) 4-15.

[3] Boehm B.W., Software Engineering Economics, Prentice-Hall, Englewood Cli4s, NJ, 1981

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

[5] Donald J. Reifer, Barry W. Boehm, Sunita Chulani, "The Rosetta Stone: Making COCOMO 81 Files Work With COCOMO II", CROSSTALK, The Journal of Defense Software Engineering, Feb 1999

[6] Harish Mittal and Pradeep Bhatia, Optimization Criteria for Effort Estimation using Fuzzy Technique, CLEI Electronic Journal, Vol 10, No 1, Paper 2, 2007

[7] Heiat A., "Comparison of artificial neural network and regression models for estimating software development effort", Information and Software Technology 44 (15), 911–922, 2002.

[8] 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

[9] Karunanithi Nachimuthu, Darrell Whitley, Yashwant K. Malaiya, "Using neural networks in reliability prediction", IEEE Software, Vol 9, Issue 4, pp. 53-59, 1992

[10] LiMin Fu, "Neural Networks in Computer Intelligence," Tata McGraw-Hill Edition 2003, pp.94-97.

[11] Mehmet Ali Yurdusev, Mahmut Firat, Mustafa Erkan Turan "Generalized regression neural networks for municipal water consumption prediction" Published in: Journal of Statistical Computation and Simulation , Vol 80, Issue 4 April 2010 , Pgs 477 - 478

[12] Prasad Reddy P.V.G.D, Sudha K.R , Rama Sree P & Ramesh S.N.S.V.S.C, Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks, Journal of Computing, Vol 2, Issue 5, May 2010, Pgs 87-92

[13] Prasad Reddy P.V.G.D, Sudha K.R , Rama Sree P & Ramesh S.N.S.V.S.C, Fuzzy Based Approach for Predicting Software Development Effort , International Journal of Software Engineering, Vol 1, Issue 1, June2010, Pgs 1-11.

[14] Ramil J.F, Algorithmic cost estimation for software evolution, Software Engg. (2000) 701-703.

[15] Zhiwei Xu, Taghi M. Khoshgoftaar, Identification of fuzzy models of software cost estimation, Fuzzy Sets and Systems 145 (2004) 141–163