Adaptive Neuro-Fuzzy Inference System (ANFIS) Based Software Evaluation

Khyati M. Mewada [1], Amit Sinhal[2] and Bhupendra Verma[3]

**[1] Dept. of Information Technology, Technocrats Institute of Technology
Bhopal, Madhya Pradesh, India**

**[2] Dept. of Computer Sc. & Engg., Technocrats Institute of Technology
Bhopal, Madhya Pradesh, India**

**[3] Dept. of Dept. of Computer Sc. & Engg., Technocrats Institute of Technology (Excellence)
Bhopal, Madhya Pradesh, India**

Abstract

Software metric is a measure of some property of a piece of software or its specifications. The goal is to obtain reproducible and quantifiable measurements, which may have several valuable applications in schedule and budget planning, effort and cost evaluation, quality assurance testing, software debugging, software performance optimization, and optimal personnel task assignments. Software effort evaluation is one of the most essential and crucial part of software project planning for which efficient effort metrics is required. Software effort evaluation is followed by software cost evaluation which is helpful for both customers and developers. Thus, efficiency of effort component of software is very essential. The algorithmic models are weak in estimating early effort evaluation with regards to uncertainty and imprecision in software projects. To overcome this problem, there are various machine learning methods. One of the methods is soft computing in which there are various methodologies viz., Artificial Neural Network, Fuzzy Logic, Evolutionary computation based Genetic Algorithm and Meta-heuristic based Particle Swarm Optimization. These methods are good at solving real-world ambiguities. This paper highlights the design of an efficient software effort evaluation model using Adaptive Neuro-Fuzzy Inference System (ANFIS) for uncertain datasets and it shows that this technique significantly outperforms with sufficient results.

Keywords: *Software metrics, Software effort evaluation, Cost evaluation, Soft Computing Techniques, COCOMO, ANFIS.*

1. Introduction

The goal of any successful software project is to develop quality software within time, cost and resource constraints. This can be achieved consistently, only through effective management of the software development process. Well-defined measures of the process and the product are necessary to exercise control and to bring about improvement in the software development process. Software metrics are quantitative measures that provide the basis for effective management of the software development process. Software metrics are used to improve software productivity and quality. “Metric is a quantitative measure of the degree to which a given attribute is possessed by a system or its component or by a process.” Software metrics are measures that are used to quantify different attributes of a software product, software development resource and software development process. *Software metrics* deals with the evaluation and measurement of different attributes of the software product and the software development process [1]. There are three kinds of software metrics: procedure metrics, project metrics, and product metrics [2].

- ✓ Procedure metrics measure the resources (time and cost) that a program development effort will take. They are useful for the administration and management of the project.
- ✓ Project metrics give information about the actual situation of the project. These metrics include costs, effort, risks, and quality. These are used to improve the development process of the project.
- ✓ Product metrics assess quality information about the program. These metrics focus on reliability, maintainability, complexity, and reusability of all or part of the software developed for the program.

The reliability of these software metrics as predictors bugs has been studied and tested by many researchers [3, 4, 5], who have used different regression models applied to different languages. All of these researchers have claimed software metrics to have good capabilities as indicators of bugs.

Metrics are seen as force multipliers in improvement initiatives and quality movements. Metrics have led organizations and individuals in a process of self-discovery of goals, capabilities and constraints. Inspired by metrics, data patterns, evaluation models for bug fixing have been constructed and as a result the bug evaluation task has been refined and redefined in many organizations. The most vital contribution of metrics is the decision-making support. Constant interpretations of metrics inject a stream of values into the organization. Problem-solving cycles have benefited from metrics in all the phases. Metrics are used for recognition and later for diagnostics of problems. Experiments are conducted to test ideas, true to the scientific spirit of metrics application.

Software effort evaluation is one of the most essential and crucial part of software project planning for which efficient effort metrics is required. Software effort evaluation is followed by software cost evaluation which is helpful for both customers and developers. Thus, efficiency of effort component of software is very essential. Software effort evaluation is an important activity in software engineering. Estimating software effort early in software development lifecycle is a challenging task. Software size estimate is one of the most important inputs for software effort evaluation. Thus providing a size estimate with good accuracy early in the lifecycle is equally important.

However, estimates that are computed early in the lifecycle are typically associated with uncertainty. To deal with this problem, many effort evaluation techniques and metrics are developed by many researchers based on many different methods. Traditionally, there are various evaluation techniques based on comparison, analogy, equations which are broadly categorized as macro evaluation techniques. There are other micro evaluation techniques also like work breakdown structure (also known as Delphi Technique) based on Expert Judgment. The other most popular method used for effort evaluation is COCOMO found by Barry Boehm in 1981. One more technique called Putnam's Life Cycle Model is also available in the literature.

2. Background

This section describes the software effort evaluation approaches, traditional effort evaluation techniques and soft computing based evaluation for effort. It also describes the software metrics.

2.1 Effort Evaluation Approaches

There are two major software evaluation approaches: macro (for example, top-down; parametric) and micro (for example, bottom-up; task based), although some evaluation approaches combine typical aspects of both macro and micro techniques. Any of the techniques could be used at any point in the life cycle. However, the more accurate is the estimate of the project's size, the more precise is the effort and duration estimates. The relative precision of resultant estimates will match the precision of inputs.

2.2 Traditional Effort Evaluation Techniques

Based on the above mentioned approaches, there are various effort evaluation techniques in both the categories.

2.2.1 Delphi Technique

When quantified or empirical data are absent, then expertise based techniques are needed. The opinion of experts is taken, but the drawback with this technique is that the estimate is as well as the expert's opinion only. For example, Delphi technique or work breakdown structure. Delphi is a place in Greece, which was supposed to confer predictive powers to the person. A temple was built there and virgin girls were appointed there to answer questions about the future, they were called oracles. Oracle's prophecies were considered prophetic or at least wise counsel [6]. So, Delphi technique was derived from them. Under this method, project specifications are given to a few experts and their opinion is taken.

2.2.2 Putnam's Life Cycle Model

The Putnam Model is an empirical software effort evaluation model [7]. Lawrence H. Putnam in 1978 [8] is seen as pioneering work in the field of Software Process Modeling. This model describes the time and effort required for a project of specified size. SLIM (Software Lifecycle Management) is the name given by Putnam. Closely related software parametric models are COCOMO (Constructive Cost Model), PRICE-S (Parametric review of Information for Costing and Evaluation Software) and (SEER-SEM) Software Evaluation and Evaluation of Resources-Software estimating model. Nordon studied the staffing patterns of several R & D projects. He noted that the staffing pattern can be approximated by a Rayleigh distribution curve. Putnam studied the work of Nordon and determined that Rayleigh curve can be used to relate the number of lines of code to estimate time and effort required by the project.

2.2.3 COCOMO

The Constructive Cost Model (COCOMO) was launched in 1981 by Barry Boehm. It is also called COCOMO 81. The model assumes that the size of a project can be estimated in thousands of delivered source instruction and then uses a non-linear equation to determine the effort for the project. COCOMO II is the successor of COCOMO 81 and is better suited for estimating modern software development projects and updated project database. The need for the new model came as software development technology moved from mainframe and overnight batch processing to desktop development, code reusability and the use of off-the-shelf software components. COCOMO consists of a hierarchy of three increasingly detailed and accurate forms. The first level, Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but its accuracy is limited due to its lack of factors to account for difference in project attributes (Cost Drivers). Intermediate COCOMO takes these Cost Drivers into account and Detailed COCOMO additionally accounts for the influence of individual project phases.

2.3 Soft Computing Based Effort Evaluation Techniques

The limitations of algorithmic models led to the exploration of the non algorithmic techniques which are soft computing based. These include artificial neural network, evolutionary computation, fuzzy logic models, case-based reasoning, and combinational models and so on.

2.3.1 Neural Networks

Neural networks are nets of processing elements that are able to learn the mapping existent between input and output data. The neuron computes a weighted sum of its inputs and generates an output if the sum exceeds a certain threshold. This output then becomes an excitatory (positive) or inhibitory (negative) input to other neurons in the network. The process continues until one or more outputs are generated [9]. It reports the use of neural networks for predicting software reliability; including experiments with both feed forward and Jordan networks with a cascade correlation learning algorithm. The Neural Network is initialized with random weights and gradually learns the relationships implicit in a training data set by adjusting its weights when presented with these data. The network generates effort by propagating the initial inputs through subsequent layers of processing elements to the final output layer. Each neuron in the network computes a nonlinear function of its inputs and passes the resultant value along its output [10]. The neural network is known for its ability in tackling the classification problem. Contrarily, in effort evaluation what is needed is generalization capability.

2.3.2 Fuzzy Logic

Fuzzy logic is a valuable tool, which can be used to solve highly complex problems where a mathematical model is too difficult or impossible to create. It is also used to reduce the complexity of existing solutions as well as increase the accessibility of control theory [11]. The development of software has always been characterized by parameters that possess a certain level of fuzziness. The study showed that the fuzzy logic model has a place in software effort evaluation [12]. The application of fuzzy logic is able to overcome some of the problems which are inherent in existing effort evaluation techniques [13]. Fuzzy logic is not only useful for effort prediction, but that it is essential in order to improve the quality of current estimating models [14]. Fuzzy logic enables linguistic representation of the input and output of a model to tolerate imprecision [15]. It is particularly suitable for effort evaluation as many software attributes are measured on nominal or ordinal scale type which is a particular case of linguistic values [16].

2.3.3 Genetic Algorithm

Genetic Algorithm is one of the evolutionary methods for effort evaluation. Evolutionary computation techniques are characterized by the fact that the solution is achieved by means of a cycle of generations of candidate solutions that are pruned by the criteria 'survival of the fittest' [17]. When GA is used for the resolution of real-world problems, a population comprised of a random set of individuals is generated. The population is evaluated during the evolution process. For each individual a rating is given, reflecting the degree of adaptation of the individual to the environment. A percentage of the most adapted individuals are kept, while that the others are discarded. The individuals kept in the selection process can suffer modifications in their basic characteristics through a mechanism of reproduction. This mechanism is applied to the current population aiming to explore the search space and to find better solutions for the problem by means of crossover and mutation operators generating new individuals for the next generation. This process, called reproduction, is repeated until a satisfactory solution is found [18].

2.3.4 Particle Swarm Optimization

Particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Such methods are commonly known as Meta Heuristics as they make little or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. An algorithm [19] is developed named Particle Swarm Optimization Algorithm (PSOA) to fine tune the fuzzy estimate for the development of software projects.

2.4 Software Metrics

"Metric is a quantitative measure of the degree to which a given attribute is possessed by a system or its component or by a process." Software metrics are measures that are used to quantify different attributes of a software product, software development resource and software development process. *Software metrics* deals with the evaluation and measurement of different attributes of the software product and the software development process. [1]

3. Literature Review

This section explores some of the researches done on software evaluation using different techniques by various researchers in previous years.

In **2010**, **Jin-Cherng Lin** et al. [20] used Pearson product moment correlation coefficient and one-way to analyze to select several factors and then used K-Means clustering algorithm to software project clustering. After project clustering, they used Particle Swarm Optimization that takes the mean of MRE (MMRE) as a fitness value and N-1 test method for optimization of COCOMO parameters. Finally, they took parameters that finish the optimization to calculate the software project effort that is wanting to evaluate. This research used 63 history software project data of COCOMO to test. The experiment really expresses using base on project clustering with multiple factors making more effective base on the effort of the software estimate of COCOMO's three project mode.

In **2011**, **Jin-Cherng** Lin et al. [21] proposed a model which combines genetic algorithm (GA) with support vector machines (SVM). We can find the best parameter of SVM regression of the proposed model, and make more accurate predictions. The model was tested and verified by using the historical data in COCOMO, Desharnais, Kemerer, and Albrecht. The results were shown by prediction level (PRED) and the mean magnitude of relative error (MMRE).

In **2012**, **Thamarai.I** et al. [22] proposed a genetic algorithm and artificial neural network based on which Feature Selection and Similarity Measure between the projects can be achieved by using Differential Evolution. This is a population based search strategy. The Differential Evolution is used to compare the key attributes between the two projects. Thus we can get most optimal projects which can be used for the evaluation of effort using the analogy method.

4. About the Problem

To get an accurate or near to accurate effort evaluation has always been a challenge in software development. To deal with this problem, many researchers have contributed in various areas by applying many techniques. These techniques include regression analysis, analogy-based evaluation, comparison based evaluation and machine learning based evaluation. The uncertainty can be reduced by any of the above techniques.

Neural networks are good at training the dataset, but the clustering and feature input is somewhat weak in neural networks. Fuzzy logic based models are good at featuring and clustering but the training of datasets is not provided. Genetic algorithms as an optimization technique are usually applied in multi-neural systems in order to improve operations or performance of the system, either as an expert or global level. PSO easily suffers from the partial optimism, which causes the less exact at the regulation of its speed and the direction. The method cannot work out the problems of scattering and optimization. The method cannot work out the problems with non-coordinate system, such as the solution to the energy field and the moving rules of the particles in the energy field.

From the above survey, it is clear that each of the methods mentioned has some of the disadvantages over the other. To overcome this problem, this paper gives the detail of an efficient framework for effort evaluation using neuro-fuzzy technique i.e., ANFIS.

5. Proposed Framework

The proposed framework includes the evaluation of software effort using neuro-fuzzy based (ANFIS toolbox) of MATLAB. The details for proposed framework are mentioned as under:

- For software effort evaluation, NASA dataset with 18 projects is considered for implementation. The performance measures MMRE and RMSE are used for comparing the performance of ANFIS in effort evaluation with other traditional evaluation models.

Thus, the framework measures the effort component of the software efficiently using ANFIS which in turn is useful for cost evaluation of software.

This paper proposes ANFIS based software effort evaluation. ANFIS is a hybrid AI technique, which combines best features of Fuzzy Logic and parallel processing neural networks. It possesses fast convergence and has more accuracy than back propagation neural network. Various forms of ANFIS methods are explored for effort evaluation. ANFIS methods are comparatively good at evaluation than complex neural networks.

The Sugeno based Fuzzy Inference system is developed and in order to train the Sugeno FIS, Adaptive Neuro-Fuzzy system (ANFIS) is designed that makes use of the Sugeno FIS Structure as shown in Fig. 1.

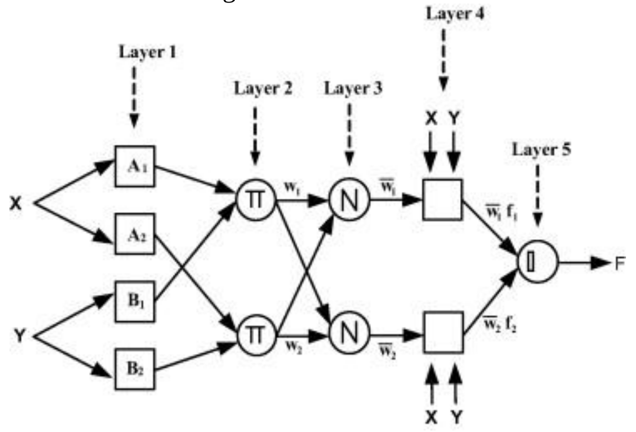

Fig.1. Architecture of ANFIS

The algorithm used for effort evaluation is based on Neuro-Fuzzy technique. More specifically, it is known as Adaptive Neuro-Fuzzy Inference System (ANFIS). The implementation is done on a NASA dataset of 18 projects in MATLAB R2011a Environment. The steps of the proposed algorithm are shown in the form of the flowchart in Figure 2.

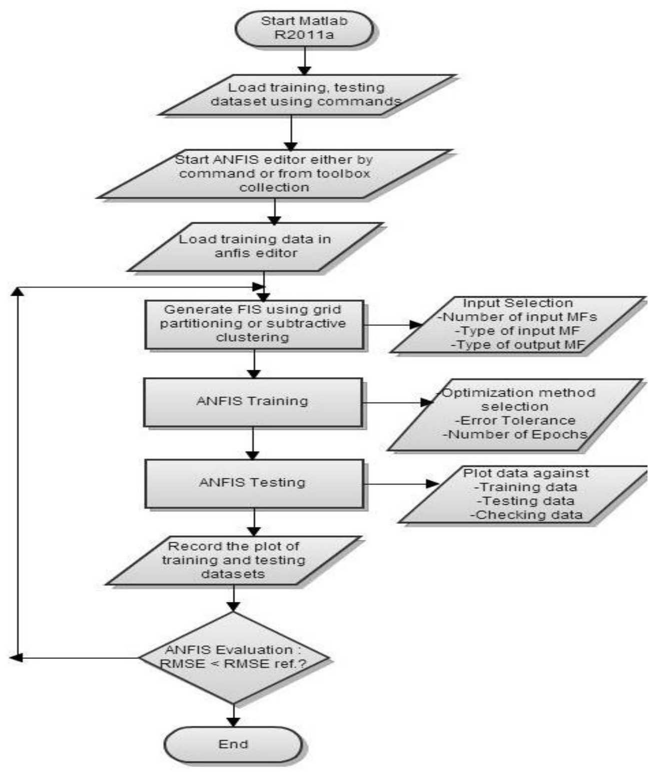
Start Matlab R2011a
Load training, testing dataset using commands
Start ANFIS editor either by command or from toolbox collection
Load training data in anfis editor
Generate FIS using grid partitioning or subtractive clustering
Input Selection -Number of input MFs -Type of input MF -Type of output MF
ANFIS Training
Optimization method selection -Error Tolerance -Number of Epochs
ANFIS Testing
Plot data against -Training data -Testing data -Checking data
Record the plot of training and testing datasets
ANFIS Evaluation : RMSE < RMSE ref.?
End

Fig.2. Flowchart of ANFIS

6. Experimentation and Results

The list of parameters used for simulation in MATLAB is shown in the Table 1 below:

Table 1: List Of Parameter Variables And Their Values

Parameter Variables	Associated Values
Simulation Tool	MATLAB 7.12.0 (R2011a)
Dataset used for experimentation	NASA dataset
Total No. of projects	18
No. of projects used for training	13
No. of projects used for testing	5
FIS method	Grid Partitioning
Optimization method	Hybrid
No. of membership functions	2
Type of membership functions	Trimf, Trapmf, gbellmf, gaussmf, gauss2mf, pimf, dsigmf, psigmf.
No. of epochs	500

In Table 2 and 3 the computed effort for training datasets is described for each membership function using ANFIS toolbox of MATLAB. In Table 4 and 5 *RMSE* and *MMRE* criteria is computed over the complete data set for ANFIS model for different membership functions is shown [23]. In Table 6 computed software quality metric for membership functions compared with the other models are shown.

Neuro-Fuzzy model using ANFIS toolbox of MATLAB uses different membership functions. There are 8 functions in ANFIS, out of which gauss2 membership function has the lowest MMRE and RMSE of 0.0050 and 0.6410 respectively. Also, gbell membership function has the lowest MMRE and RMSE of 0.0367 and 0.4976.

The software quality metric EEA should approach to 1 and SP (total) should be equal to the value resulting from total source size divided by the actual effort value. Here, in this case, we have taken the comparison for a first project from datasets of all projects. Accordingly, the value of EEA and SP (total) for neuro-fuzzy functions outperforms the other traditional models.

Table 2: Computed Effort For Nasa Software Projects-Training Case - ANFIS Functions

No.	DKLOC	Method-ology	Actual Effort	Trimf Effort	Trapmf Effort	Gbellmf Effort	Gaussmf Effort
1	90.2	30	115.8	115.7801	115.7990	115.8005	115.7801
2	46.2	20	96	95.6446	88.6000	95.9679	95.6446
3	46.5	19	79	79.2911	88.6000	79.0235	79.2911
4	54.5	20	90.8	90.7932	88.6000	90.8010	90.7932
5	31.1	35	39.6	39.8002	39.6113	39.6464	39.8002
6	67.5	29	98.4	98.4448	98.4015	98.3999	98.4448
7	12.8	26	18.9	19.9137	10.2485	18.9679	19.9137
8	10.5	34	10.3	9.4786	8.8195	9.2436	9.4786
9	21.5	31	28.5	27.9643	28.4680	28.4765	27.9643
10	3.1	26	7	5.7052	10.2485	6.2696	5.7052
11	4.2	19	9	9.0087	9.0000	9.0504	9.0087
12	7.8	31	7.3	8.4028	8.9545	8.4367	8.4028
13	2.1	28	5	5.3722	10.2485	5.5157	5.3722

Table 3: Computed Effort For Nasa Software Projects-Training Case - ANFIS Functions

No.	DKLOC	Method-ology	Actual Effort	Gauss2mf Effort	Pimf Effort	Dsigmf Effort	Psigmf Effort
1	90.2	30	115.8	115.7970	115.7995	115.7972	115.7972
2	46.2	20	96	95.9509	88.6000	95.9353	95.9334
3	46.5	19	79	79.0113	88.6000	79.0390	79.0396
4	54.5	20	90.8	90.8375	88.6000	90.8235	90.8247
5	31.1	35	39.6	39.6263	39.6083	39.6210	39.6154
6	67.5	29	98.4	98.4053	98.4008	98.4048	98.4048
7	12.8	26	18.9	18.8415	10.2625	18.8965	18.8706
8	10.5	34	10.3	9.0406	8.8123	9.0812	9.0714
9	21.5	31	28.5	28.4568	28.4770	28.4656	28.4914
10	3.1	26	7	6.0100	10.2625	6.0194	6.0301
11	4.2	19	9	9.0034	9.0000	9.0039	9.0040
12	7.8	31	7.3	8.6562	8.9143	8.7286	8.7210
13	2.1	28	5	5.9624	10.2625	5.7833	5.7953

Table 4: Computed RMSE And MMRE Criterion For Anfis Functions

Performance Criteria	ANFIS Function Used			
	trimf	trapmf	gbellmf	gaussmf
MMRE	0.1770	0.1974	0.0367	0.0443
RMSE	0.6369	4.5543	0.4976	0.6369

Table.5 Computed RMSE And MMRE Criterion For Anfis Functions

Performance Criteria	ANFIS Function Used			
	gauss2mf	pimf	dsigmf	psigmf
MMRE	0.0050	0.1973	0.0472	0.0473
RMSE	0.6410	4.5533	0.6269	0.6269

Table 6: Computed EEA and SP (total) Criterion for ANFIS Functions for project 1 in the dataset

Software quality metric	trimf	trapmf	gbellmf	gaussmf	gauss2mf	pimf	dsigmf	psigmf
EEA	1	1	0.9999	1.0001	1	1	1	1
SP (total)	0.799	0.7789	0.7789	0.779	0.7789	0.779	0.7789	0.7789

7. Conclusion and Future Work

This paper evaluates software effort efficiently using ANFIS based learning techniques. Accurate evaluation of effort leads to other evaluations efficiently and accurately like cost, staffing, budget and schedule. Effort component of software plays a vital role in software project management. By predicting software effort, proposed ANFIS based technique may facilitate the software planning stage in making its decision regarding the evaluation of other resources of the software. The ANFIS based technique was successfully implemented to predict software effort. The same was compared with neural network based technique and other models which were previously reported. In the basic scheme three types of ANFIS were used for learning. All three provided better performance in all performances metric with respect to the neural network. Conventional ANFIS worked better in accuracy and RMSE error compared all type neural networks and other previous methods.

Software effort doesn't only depend on the size of the project, it may include different other parameters like Intermediate COCOMO attributes. So ANFIS based technique must be tuned to predict all these attributes affecting software effort. So for these different types of attributes hybrid ANFIS must be explored. In the extended case only the methodology and size of the project parameters are included but practical situations also affect effort and other resources. The problem must be formalized to include other parameters which affect effort. The prediction was based on an assumed scenario but to validate and check the robustness of ANFIS more realistic time series must be considered for training.

1. Analysis can be made for another type of datasets like ISBSG, IBM etc.
2. Calculation of Pred (25), Spearman’s rank can lead to better validation of prediction models.
3. Analysis can also be done using artificially generated data set.
4. Analyzing the performance of the model by varying the number of epoch, number of membership functions.

References

[1] Jibitesh Mishra, Ashok Mohanty, (2011) “Software Engineering”, CH-13, Software Metrics, Pearson Education India.

[2] Tu Honglei, Sun Wei, and Zhang Yanan, “The research on software metrics and software complexity metrics”, in Computer Science-Technology and Applications, 2009. IFCSTA ’09. International Forum, volume 1, pages 131 –136, Dec. 2009.

[3] S.R. Chidamber and C.F. Kemerer. “A metrics suite for object oriented design”, IEEE Transactions on Software Engineering, 20(6):476 –493, Jun. 1994.

[4] M. D’Ambros, M. Lanza, and R. Robbes. An extensive comparison of bug prediction approaches. In 7th IEEE Working Conference on Mining Software Repositories (MSR), 2010, pages 31 –41, May 2010.

[5] V.R. Basili, L.C. Briand, and W.L. Melo. “A validation of object-oriented design metrics as quality indicators”, IEEE Transactions on Software Engineering, 22(10):751 –761, Oct. 1996.

[6] Murali Chemuturi, Delphi Technique for software evaluation.

[7] Putnam, Lawrence H.; Ware Myers (2003). Five core metrics: the intelligence behind successful software management. Dorset House Publishing. ISBN 0-932633-55-2.

[8] Putnam, Lawrence H. (1978). "A General Empirical Solution to the Macro Software Sizing and Estimating Problem". IEEE transactions on Software Engineering, VOL. SE-4, NO. 4, pp 345-361.

[9] N. Karunanitthi, D. Whitley, and Y. K. Malaiya, (1992), "Using Neural Networks in Reliability Prediction”, IEEE Software, Vol. 9, no.4, pp. 53-59.

[10] Idri, T. M. Khoshgoftaar, A. Abran, (2002), “Can neural networks be easily interpreted in software cost evaluation?” IEEE Trans. Software Engineering, Vol. 2, pp. 1162 – 1167.

[11] Razaz, M. and King, J. (2004)”Introduction to Fuzzy Logic” Information Systems - Signal and Image Processing Group. http://www.sys.uea.ac.uk/king/restricted/boards/.

[12] Moon Ting Su1, Teck Chaw Ling, Keat Keong Phang, Chee Sun Liew, Peck Yen Man, (2007), "Enhanced Software Development Effort And Cost Evaluation Using Fuzzy Logic Model", Malaysian Journal of Computer Science, Vol. 20(2), pp 199-207.

[13] R. Gray, S. G. MacDonell, (1997), “Applications of Fuzzy Logic to Software Metric Models for Development Effort Evaluation”, Fuzzy Information Processing Society 1997 NAFIPS’ 97, Annual Meeting of the North American 21 - 14 pp. 394-399.

[14] S. Kumar, B. A. Krishna, and P. S. Satsangi, (1994), “Fuzzy systems and neural networks” in software engineering project management, Journal of Applied Intelligence, no. 4, pp. 31-52.

[15] N. E. Fenton, S. L. Pfleeger, (1997),”Software Metrics, a Rigorous and Practical Approach”, 2nd Edition, PWS Publishing Company, Thomson Publishing,Boston.

[16] Agustin Gutierrez T., Cornelio Yanez M. and Jerome Leboeuf Pasquier, (2005), “Software Development Effort Evaluation Using Fuzzy Logic: A Case Study”, Proceedings of the Sixth Mexican International Conference on Computer Science (ENC’05).

[17] Urkola Leire , Dolado J. Javier , Fernandez Luis and Otero M. Carmen , (2002), "Software Effort Evaluation: the Elusive Goal in Project Management", International Conference on Enterprise Information Systems, pp.412-418.

[18] A.P. Engelbrecht, (2006), Fundamentals of Computational Swarm Intelligence, John Wiley & Sons, New Jersey.

[19] Prasad Reddy, (2010), “Particle Swarm Optimization in the fine-tuning of Fuzzy Software Cost Evaluation Models, International Journal of Software Engineering (IJSE), Volume (1): Issue (1), pp 12-23.

[20] Jin-Cherng Lin, Han-Yuan Tzeng (2010), "Applying Particle Swarm Optimization to Estimate Software Effort by Multiple Factors Software Project Clustering", Computer Symposium (ICS), Proceedings published in IEEE, pp. 1039-1044.

[21] Jin-Cherng Lin, Chu-Ting Chang and Sheng-Yu Huang (2011), "Research on Software Effort Evaluation Combined with Genetic Algorithm and Support Vector Regression", International Symposium on Computer Science and Society, Proceedings published in IEEE, pp. 349-352.

[22] Thamarai.I, Dr.S.Murugavalli (December 2012), "Using Differential Evolution in the Prediction of Software Effort", IEEE Fourth International Conference on Advanced Computing (ICoAC), pp. 1-3.

[23] http://www.mathworks.in/help/fuzzy/anfis-and-the-anfis-editor-gui.html.