Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Quality Prediction in Object Oriented System by Using ANN: A Brief Survey

Yajnaseni Dash*,Sanjay Kumar Dubey
Department of Computer Science & Engineering,
Amity School of Engineering & Technology,
Amity University

***Abstract*- At present quality of software systems is a major issue, still well defined criteria to measure it needs to be established. The object-oriented (OO) systems, which is different from procedural paradigm requires valid and effective metrics to assess quality of the software. There is considerable research interest in developing and applying sophisticated techniques to construct models for estimation. The different soft computing techniques such as Artificial Neural Networks (ANN), fuzzy inference systems and adaptive neuro-fuzzy inference systems are available for the prediction purpose. However among these techniques, ANN possesses the advantages of predicting software quality accurately and identifies the defects by efficient discovery mechanisms. This paper aims to survey various research methodologies proposed to predict quality of OO metrics by using neural network approach.**

***Keywords*— software quality, maintainability, object oriented, cost estimation, artificial neural network**

I. Introduction

Currently software quality is a major factor of concern. The growing research activity in software quality leads to innovation of novel techniques, to predict its attributes. Maintainability is an important quality attribute and a difficult concept as it involves a number of measurements. OO metrics are used in quality estimation. However quality estimation means estimating maintainability or reliability of software. Software reliability is a valuable ingredient to make the system work properly without a fail [1]. As the OO system uses a huge amount of small methods, it is time consuming, error prone and has a distinctive maintenance problem [2]. ANN is one such technique which involves a series of steps for the computation of maintainability effort. As the traditional computers are not excellent to interact because of the noised data, immense parallelism, fault tolerant, and failure to adapt to certain circumstance, so ANN provides a better option for handling software quality. The neural network system also helps us when we are unable to devise an algorithmic elucidation. The application of ANN for software quality prediction using object-oriented metrics is focused in this paper.

II. NEURAL NETWORK

The Neural network is a network of interconnected neurons where information propagation occurs by firing electrical pulses via its connections. During the lifetime of a neuron, the connections or weights need to be adjusted. Similarly the quantity of incoming pulses which required in activating a neuron is changed. This is the behavioural attributes that lets the NN to learn. There are three basic learning algorithms in neural networks namely supervised learning, unsupervised learning and reinforced learning. Supervised learning is most commonly used and also called as learning with a teacher. It is applied when the target value is known and associated with each input in the training set.

A. *History of neural network*

An ANN is an information processing paradigm that is an emulation of biological neural system. A trained neural network can be thought of as an expert in the category of information it has been given to analyse.

The first essential model of neural network was proposed by McCulloch and Pitts [3]. It is a computing model of activities of nervous system and acts as a binary device where every neuron has predetermined threshold logic. Various researchers (e.g. Jhon von Neumann, Marvin Minsky, Frank Rosenblatt) worked on this model.

Hebb [4] described in his standard book “The Organization of Behaviour”, the proper interconnection and self organization of neurons. The existing path which lies between the neurons strengthens the connections. As an enormous number of simple neurons embedded in an interactive nervous system, it is promising to provide computational power for extremely complicated information processing. The basic learning rule in neural network literature is Hebb’s law. The perceptron model was developed by Rosenblatt [5] which follows perceptron learning law. ADALINE (Adaptive Linear Element) for computing element and LMS (Least mean square) learning algorithms explained by Windrows and hoff [6] for adjusting the weights of an ADALINE model. Rumelhart and McClelland [7] illustrated the weight of multilayer feed forward neurons can be adjusted probably in a logical approach by mapping them implicitly in a set of input and output pattern pairs for learning purposes. This act of learning is named as generalized delta rule or error back propagation rule. A marvellous outline of ANN was suggested by Cheng and Titterington [8] and Warner and Mishra [9]. Zang *et al.* [10] provided the broad-spectrum procedures for network modelling for the use of forecasting purpose by using statistical techniques. The performance of multilayer perceptron and linear regression models for the purpose of prediction of quality was also compared. There are various commercial and free software are available which has been made for the prediction of faults in the neural networks.

B. Neural Network Architecture

Neural network architecture refers to the types of interconnections between neurons. A network is said to be fully connected if the output from a neuron is connected to every other neuron in the next layer [11].

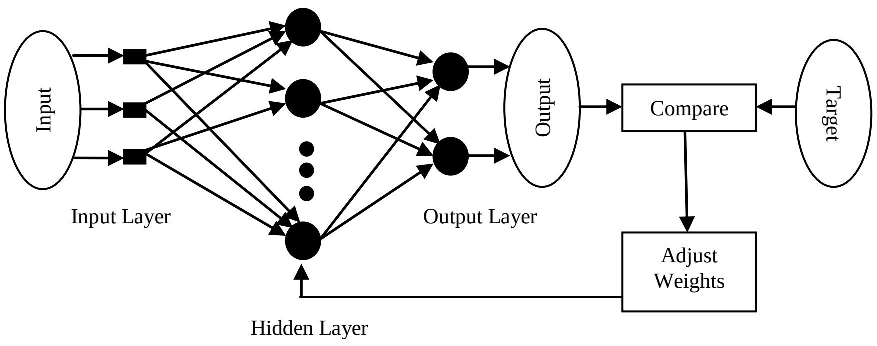

Fig 1 A sample architecture of ANN

There are three layers in an ANN such as input layer, hidden layer and output layer as shown in Fig 1.

1) Input Layer: This layer comprises of the input units which symbolizes the unrefined information provided for the network.

2) Hidden Layer: This layer is represented by hidden units which are influenced by the behaviour of the input units and the weights that connect these input and the hidden units.

3) Output Layer: The output unit’s behaviour is dependent on the specificity of the hidden units and the weights connecting the hidden and output units.

Generally the ANN adjusts the values of the weights to produce a specific target output from a particular input. Then it compares the output with the target till a match between the target and the output is found. A neural network can be trained to do a specific function only by taking several such input or target pairs into consideration.

C. Connection structures of neural network

A neural network comprises the neuron and weight building blocks. The behaviour of the network depends largely on the interaction between these building blocks. There are four types of weighted connections namely feed forward, feedback, lateral, and time-delayed connections [12-13].

1) Feed forward connections: For all the neural models, data from neurons of a lower layer are propagated forward to neurons of an upper layer via feed forward connections networks. It is a one way communication process where the signal flows from input to output. Multilayer feed forward network is also known as multilayer perceptron.

2) Feedback Connections: Feedback networks bring data from neurons of an upper layer back to neurons of a lower layer. It is a two way communication process where signal flows in a bidirectional manner. Symmetric recurrent neural network are also known as attractor or hopfield neural networks.

3) Lateral Connections: One typical example of a lateral network is the winners-takes-all circuit, which serves the important role of selecting the winner. In the feature map example, by allowing neurons to interact via the lateral network, a certain topological ordering relationship can be preserved.

4) Time-delayed Connections: Delay elements may be incorporated into the connections to yield temporal dynamics models. They are more suitable for temporal pattern recognitions.

The synaptic connections may be fully or locally interconnected. Also, a neural network may be either a single layer feedback model or a multilayer feed-forward model. It is possible to cascade several single layer feedback neural nets to form a larger net.

III. ANALYSIS OF NNS IN SOFTWARE COST ESTIMATION

Computation of software effort is a challenging task as the raw data available during initial parts of project development are inconsistent, haphazard and incomplete. All the estimation models are greatly used to predict the software effort based on the mathematical formula of effort such as, effort = α × size β. Besides this, various techniques like ANN, analogy based reasoning, regression trees, rule based induction models are also considered for the estimation of software effort. Idri *et al.*

(2002) explored cost estimation models based on ANN. They had used feed forward multilayer perceptron as architecture, back-propagation as learning algorithm and selected sigmoid function as the activation function of ANN. Various studies has been done to estimate software effort accurately by applying NN approach. Prasad *et al.* [15] compared software effort estimation by using radial basis and generalized regression neural networks and found that radial basis neural network is more suitable.

A. *Advantages and disadvantages*

Use of ANN for prediction of software effort possesses several advantages; still there are causes of not preferring the ANN technique for software cost estimation. Some of these are presented in Table 1.

TABLE I
PROS AND CONS OF ANN IN EFFORT PREDICTION

S. No.	Advantages	Disadvantages
1	Software development is incessantly emerging process. 'Learning from the past situations' is the most important criteria of ANN, which acts like a path finder for future development of software.	NN approach is just as like as the black box technique where we are unable to understand the internal structure of the network.
2	It is applicable for developing efficient algorithms in the fields where little knowledge is available.	It can solve problems comprising high complexity such as classification and categorization type of problems. However in case of cost estimation generalization is taken into consideration rather than classification.
3	It can derive precious information and regularities from large databases inherently.	No guidelines are available to construct the NN topologies.
4	It can work efficiently where large numbers of variables are present and the conditions are changing frequently.	Decision making in ANN is a difficult task, still it is used by researchers for its learning mechanism.
5	ANN can correlate composite relationship between dependent variable (e. g. maintenance effort) and independent variable (e. g. software metrics).	

IV. LITERATURE SURVEY

Since the last decade a lot of object-oriented metrics have been proposed. These metrics can be employed to obtain assurances about software quality by the usage of prediction models. Maintainability is characteristically measured as change effort which can either be the average effort for making a change to a class, or the overall effort used on changing a class.

In the literature, a number of metrics suggested to emphasize on the quality of object-oriented (OO) design by various researchers. To capture the distinctive facet of OO design, various metrics designed which estimate the quality of the software. These metrics are very fast in assessing the bulky software with little maintenance cost and can be used in earlier phases of software development. However it is necessary to know the metrics that are valuable in capturing significant quality attributes, e.g. effort, productivity, fault-proneness and evaluation of maintainability. The relation between maintainability and metrics can be discovered by using two techniques; either objective or subjective. The metrics theoretically and empirically validated according to the previous studies which evaluates the impact of the metrics on maintainability. Then after predictive models are constructed using ANN like soft computing techniques to estimate the quality attributes of the software system.

OO methodology has been enormously utilized in software engineering. As its use increases rapidly, the current applications need to be enhanced for the further development process. Li and Henry [16] have constructed a prediction model integrating ten object oriented metrics with the help of the statistical analyses technique. The result of their study has established a strong relationship between metrics and maintenance effort in OO systems. Basili *et al.* [17] obtained that the Chidamber and Kamerer metrics were associated with fault proneness. It was based on a study of eight medium-sized systems devised by students. Khoshgaftaar *et al.* [18] introduced prediction of software quality by the use of the neural networks as a tool. A large telecommunications system has been presented by them which classify the modules as either fault prone or not fault prone. They had made a comparison between the ANN model and a non-parametric discriminate model which shows that the ANN model had better predictive accuracy than the other one. Fenton and Neil [19] evaluated various software defect prediction models, size and complexity metrics to predict defects. They disagree on the theory of "software decomposition" in order to test hypotheses about defect introduction and to help for construction of an improved science of software engineering. They made comparison of fault-proneness estimation models and concluded that software quality is a vital prerequisite in the system development.

Giovanni [20] correlated a set of static metrics and software fault-proneness. Statics metrics statically computed on the source code (e.g. Mccabe's cyclomatic complexity) whereas dynamic metrics measure thoroughness of testing (e.g.

structural and dataflow coverage). Such metrics partially influences the software fault- proneness and provides limited support for adjusting the testing process. Khoshgoftaar *et al.* [21] demonstrated a case study of real time avionics software that could predict the testability of each module by doing static measurements of source code. They observed that ANN is an ensuring technique for constructing predictive models, as they are capable of modelling nonlinear relationships. Emam *et al.* [22] have constructed a model to predict which classes in a future release of a commercial Java application will be faulty. The model was then validated on a subsequent release of the same application. Their results indicated that the prediction model had a high accuracy. Fioravanti and Nesi [23] have extracted over 200 different OO metrics to identify a suitable model for detecting fault-proneness of classes. They concluded that only little of them were appropriate for identifying fault-prone classes. Genero *et al.* [24] have presented a set of metrics to measure the structural complexity of UML class diagrams and to use them for predicting their maintainability. This will be correlated with object oriented information system maintainability to a great extent. Briand *et al.* [25] investigated the relationship between OO metrics and the probability of fault detection in system classes empirically. They studied a set of OO metrics in terms of their effectiveness in predicting fault-proneness and empirically validated these metrics as an important software quality indicator. Validation is carried out by them using two kinds of data analysis techniques such as regression analysis and discriminate analysis. They researched about the relationships between existing OO coupling, cohesion, inheritance measures and the probability of fault detection in system classes during testing through empirical observations. Their univariate analysis has shown that many coupling and inheritance measures are strongly related to the probability of fault detection in a class. Their multivariate analysis results showed that by using some of the coupling and inheritance measures, very accurate models could be derived to predict in which classes most of the faults actually lie.

Quah and Thwin [26] predicted the number of faults in a particular class by using a multiple regression model and a neural network model. They have used three industrial real-time subsystems data and found that neural network model can predict more accurately than regression model. Yu *et al.* [27] choose eight metrics to examine the relationship between these and the fault-proneness of the software systems. The subject system was the client side of a large network service management system developed by three professional software engineers. This application was written in Java comprising of 123 classes and about 34,000 lines of code. First of all, they tested the correlation between the metrics and got four extremely correlated subsets. Subsequently, they employed univariate analysis to specify the metrics that could detect faults and the one which could not. Menzies *et al.* [28] have compared various methods such as decision trees, naïve Bayes, and 1-rule classifier by using the NASA software defect data. Mahaweerawat [29] proposed fault prediction model based on supervised learning using multilayer perceptron network and the results are analysed in terms of classification. Faulty classes are again analysed and classified according to the particular fault.

Gyimothy *et al.* [30] has done empirical validation of Chidamber and Kamerer [31] metrics on open source software to predict the fault. Regression (linear and logistic regression) and machine learning methods (neural network and decision tree) were employed by them to construct models. Bellini [32] compared fault-proneness estimation models by applying the logistic regression and the discriminant analyses methodologies. They have used various datasets according to the requirement. Tian and Noore [33] applied evolutionary neural network modelling to predict software cumulative failure time. Thwin and Quah [34] applied neural networks for software quality prediction. They have taken OO metrics as independent variable and used two neural network models namely ward neural network and general regression neural network (GRNN). They concluded that GRNN network model can predict more accurately than Ward network model.

Yan Ma [35] explored exact prediction of fault prone modules in the software improvement process enables effective invention and recognition of defects. Sandhu *et al.* [36] compared all the classes of WEKA’s machine learning algorithms and experimented that Logistic Model Trees algorithm is most excellent prediction techniques among other classes of machine learning algorithms in prediction of severity of faults in software systems. Hu and Zhong [37] proposed a model to predict software module risk based on neural network. Thy applied the learning vector quantization network for prediction of the software quality. Zhao and Zhong et al [38] suggested a software fault-proneness prediction model by support vector machine and the Chidamber-Kemerer (C&K) object-oriented metrics. Yogesh *et al.* [39] predicted the testing effort of software quality attribute by applying ANN method and taking CK’s OO metrics as the independent variable. They used the public domain data of NASA and estimated the testing effort. Sandhu *et al.* [40] had explored the impact of faults in OO software modules. The primary objectives of their work are discovering the structural code and design attributes of software systems and getting the best algorithms which can be employed to model impact of faults in OO software. They also predicted the impact of the defect on the overall environment in function based systems. They observed that neuro-fuzzy based predictor models are the best one. They used public domain defect dataset of NASA and coded it in C programming language.

Arvindar *et al.* [41] predicted the software maintenance effort by applying various soft computing approaches. They used the data of two commercial software products and observed that these soft computing techniques are more useful for the construction of accurate models to predict the maintenance effort. They have chosen maintenance effort as dependent variable and eight OO metrics as independent variable. Ratra *et al.* [42] compared early prediction of fault prone modules in software systems by using clustering and neural network techniques. They measured the performance of the two approaches based on their accuracy, the mean absolute error and root mean square error values. They found that the performance of neural network approach is much better than clustering based approach.

Several measures have been proposed by researchers to capture the quality of OO code and design and used for identifying fault-proneness of classes. Various investigations also had made to predict software quality by using statistical methods. ANN has seen an explosion of interest over the years, and is being successfully applied across a range of problem domains such as finance, medicine, engineering, geology and physics. Indeed, if there are problems of prediction, classification or control, neural networks will be an efficient asset to solve them.

V. Conclusions

Researchers acknowledged that software quality prediction has a significant role in designing the object oriented software. To get over the issues of predicting maintainability in OO system, suitable and efficient metrics must be defined and chosen. The application of artificial neural networks will be an efficient method to estimate maintainability in object oriented system. Among the different soft computing techniques ANN possesses advantages of predicting the software maintenance effort by minimal computation. It can be used as a predictive model because of its incredible representation techniques and ability to perform complicated functions. In addition, it is among one of the emerging technique in software development and plays a crucial role in predicting software quality.

References

[1] M. R. Lyu, Handbook of software Reliability Engineering IEEE Computer Society Press, McGraw Hill, 1996.

[2] R. E. Johnson and B. Foote Designing Reusable Classes. Journal of Object-Oriented Programming, vol. 1, no. 2, pp. 22-35, 1988.

[3] W.S. McCulloch and W. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, Vol 5, pp 115-133, 1943. Reprinted in Anderson & Rosenfeld, pp 18-28,1988.

[4] D.O. Hebb, The organization of behaviour – a neurophysiological theory. Wiley, New York, 1949.

[5] F. Rosenblatt, The Perception: A Probabilistic Model for Information Storage and Organization in the Brain, Psychological Review, ISSN 0033-295X, Vol. 65, No. 6, pp. 386-408, 1958.

[6] B. Widrow and M.E. Hoff "Adaptive Switching Circuits," IRE WESCON Convention Record, New York, pp. 96-104, 1960.

[7] D. E. Rumelhart, and J. L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1 (Foundations), The MIT Press, ISBN 0-262-68053-x, Cambridge, Massachusetts, 1986.

[8] B. Cheng and D.M.. Titterington Neural networks: A review from a statistical perspective. Statistical Science, vol 9, pp.2-54. 1994

[9] B. Warner and M. Misra, "Understanding neural networks as statistical tools", The American Statistician, vol 50, issue 4, pp.284-293, 1996.

[10] G.P. Zhang, E.P. Patuwo and M.Y. Hu, "Forecasting with artificial neural networks: The state of the art", International Journal of Forecasting, vol 14, pp. 35-62, 1998.

[11] F. Nielsen, Neural Networks – algorithms and applications. 2001 Available at: http://www.glyn.dk/download/Synopsis.pdf. Accessed on Dec.2011.

[12] Available: at: http://www.gc.ssr.upm.es/inves/neural/ann1/concepts/concepts.htm. Accessed on Dec.2011.

[13] Available: Wikipedia (2011), www. wikipedia.org, 2011

[14] 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, pp.1162–1167. 2002.

[15] P.V.G.D Prasad Reddy, K.R. Sudha, S. P. Rama and S.N.S.V.S.C Ramesh, "Software Effort Estimation using Radial Basis and Generalized Regression NeuralNetworks", Journal Of Computing, Volume 2, Issue 5, May 2010

[16] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability", Journal of Systems and Software, vol 23, no.2, pp.111-122, 1993.

[17] V. Basili, L.Briand and W. Melo "A Validation of Object-Oriented Design Metrics as Quality Indicators", IEEE Transactions on Software Engineering, vol. 22 no.10, pp. 751-761, 1996

[18] T.M. Khoshgaftaar, E.D. Allen, J.P. Hudepohl and S.J. Aud "Application of neural networks to software quality modeling of a very large telecommunications system," IEEE Transactions on Neural Networks, Vol. 8, No. 4, pp. 902--909, 1997.

[19] N. E. Fenton, and M. Neil, (1999), "A Critique of Software Defect Prediction Models", Bellini, I. Bruno, P. Nesi, D. Rogai, University of Florence, IEEE Trans. Softw. Engineering, vol. 25, Issue no. 5, pp. 675-689.

[20] D. Giovanni (2000), "Estimating Software Fault-Proneness for Tuning Testing Activities" Proceedings of the 22nd International Conference on Software Engineering (ICSE2000), Limerick, Ireland, Jun.2000.

[21] T. M. Khoshgoftaar, E. B. Allen, Zhiwei Xu, "Predicting testability of program modules using a neural network", In Proc. of 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, pp.57-62, 2000.

[22] E.L. Emam, W. Melo and C.M. Javam "The Prediction of Faulty Classes Using Object-Oriented Design Metrics", Journal of Systems and Software, Elsevier Science, pp. 63-75, 2001.

[23] F. Fioravanti and P. Nesi "A study on fault-proneness detection of object-oriented systems", Fifth European Conference on Software Maintenance and Reengineering, pp. 121 –130, 2001.

[24] M. Genero, J. Olivas, M. Piattini and F. Romero ""Using metrics to predict OO information systems maintainability", Proceedings. of the 13th International Conference Advanced Information Systems Engineering, Interlaken, Switzerland, 2001.

[25] L. Briand, W.L. Melo and J. Wust "Assessing the applicability of fault-proneness models across object-oriented software projects", IEEE Transactions on Software Engineering, vol. 28 pp. 706 –720, 2002.

[26] J. T. S. Quah, M. M. T. Thwin, "Prediction of Software Readiness Using Neural Network", In Proceedings of 1st International Conference on Information Technology & Applications, Bathurst, Australia, pp. 2312-2316, 2002.

[27] P. Yu, T. Systa and H. Muller, "Predicting fault proneness using OO metrics. An industrial case study", Proceedings. of 6th European Conference on Software Maintenance and Reengineering, pp. 99 –107, 2002.

[28] T. Menzies, K. Ammar, A. Nikora, and S. Stefano, "How Simple is Software Defect Prediction?", Journal of Empirical Software Engineering, Oct.2003.

[29] A. Mahaweerawat, "Fault-Prediction in object oriented software's using neural network techniques", Advanced Virtua and Intelligent Computing Center (AVIC), Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok, Thailand, pp.1-8, 2004.

[30] T. Gyimothy, R. Ferenc and I. Siket, "Empirical validation of object oriented metrics on open source software for fault prediction", IEEE Trans. Software Engineering, vol. 31, Issue 10, pp.897 – 910, Oct.2005.

[31] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design.” IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476–493, 1994.

[32] P. Bellini, “Comparing Fault-Proneness Estimation Models”, 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05), pp. 205-214, 2005.

[33] L. Tian and A. Noore, “Evolutionary neural network modelling for software cumulative fault prediction”, Reliability Engineering & system safety, vol. 87, pp. 45-51, 2005.

[34] M. M. T. Thwin,T. S. Quah, “Application of neural networks for software quality prediction using Object-oriented metrics”, Journal of systems and software, Vol.76, No.2, pp.147-156, 2005.

[35] Y. Ma, L. Guo, “A Statistical Framework for the Prediction of Fault-Proneness”, West Virginia University, Morgantown, 2006.

[36] Sandhu, Parvinder Singh, Sunil Kumar and Hardeep Singh, “Intelligence System for Software Maintenance Severity Prediction”, Journal of Computer Science, Vol. 3, issue 5, pp. 281-288, 2007

[37] Q. Hu and C. Zhong, “Model of predicting software module risk based on neural network”(in Chinese), Computer Engineering and Applications, Vol.43, No.18, pp.106-110, 2007.

[38] Y. Zhao, C. Zhong, Z. Li, T. Yan, “Object-Oriented Software Fault-Proneness Prediction Using Support Vector Machine” (in Chinese), Computer Engineering & Science, Vol.30, No.11, pp.115-117, 2008.

[39] Y. Singh, A. Kaur, R. Malhotra,” Predicting Testing Effort using Artificial Neural Network,” Proceedings of the World Congress on Engineering and Computer Science, WCECS, San Francisco, USA, pp. 22 - 24, Oct.2008.

[40] P.S. Sandhu, U. Malhotra, and E. Ardil, “Predicting the Impact of the Defect on the Overall Environment in Function Based Systems”, World Academy of Science, Engineering and Technology, 2009

[41] A. Kaur, K. Kaur and R. Malhotra, “Soft Computing Approaches for Prediction of Software Maintenance Effort,” International Journal of Computer Applications, Vol. 1, no.16, 2010.

[42] R. Ratra, N.S. Randhawa, P. Kaur, G. Singh,” Early Prediction of Fault Prone Modules using Clustering Based vs. Neural Network Approach in Software Systems,” IJECT Vol. 2, Issue 4, Oct . –Dec. 2011