ESTIMATING EFFORT FOR CORRECTIVE SOFTWARE MAINTENANCE USING NEURALNETWORKS AND REGRESSION TECHNIQUE

Dr. Satwinder Singh,[1]RamanpreetKaur[2]
[1]AssistantProfessor, [2]Research Scholer
[1,2]Department of Computer Science Engineering &Information Technology, Baba Banda Singh Bahadur Engineering College, FatehgarhSahib, (Punjab),India
[1]satwinder.singh@bbsbec.ac.in,[2]erramancse@gmail.com

***ABSTRACT*- Software maintenance is an integral part of a software life cycle. Software maintenance is defined as the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment. Effort estimation at the early stage of software development is very difficult because of lot of uncertainty in input parameters which decides the software effort. The objective is to modify existing software product while preserving its integrity. The study aiming at constructing cost estimation models for corrective maintenance projects. The resulting models, constructed using multivariate linear regression techniques and neural network techniques. The neural network was trained and validated for various feedforward backprop training algorithms available in Matlab NN toolbox. A class of evaluation criteria includes the PRESS statistics, SPR, MMRE, MdMRE, $PRED_{25}$, $PRED_{50}$ and neural network training and learning functions (trainlm, traingdm, learngd , learngdm,) .**

***KEYWORDS*- Software maintenance, KLOC, Effort estimation, Multiple linear regression, software development, artificial neural networks, feed forward neural networks .**

I. INTRODUCTION

The main goal of software project cost and effort estimation is to scientifically estimate the required workload and its corresponding costs in the life cycle of software system. Software cost estimation is a complex activity that requires knowledge of a number of key attributes that affect the outcomes of software projects, both individually and in concert [1]. The system changes due to corrective and non-corrective software actions.

Software maintenance can be required for four main reasons as follows[1],[2],[3]:

- Corrective Maintenance of a software product is necessary to remove errors and bugs observed while the system is in use.
- Adaptive maintenance is necessary to run software product on new platforms, on new operating systems, or when they need the product to interface with new hardware or software.
- Perfective maintenance is to enhance the system by improving efficiency, reliability and functionality.
- Preventive maintenance is to anticipate problems and correct them before they occur. Files and databases must be updated periodically, reorganized and backed up regularly.

The major issues in estimation related to software maintenance efforts include the software system's size and maintenance productivity.
Corrective maintenance is the reactive modification of a software product performed after delivery to correct discovered faults. Studies conducted by different researchers reveal that approximately 50 to 75% of the effort is spent on maintenance, out of which about 17 to 21% is exercised on corrective maintenance.
There are some sizing approaches for estimating the software maintenance efforts such as source lines of code (SLOC) , function points (FPs) , and object points [2],[3] .The annual change traffic (ACT) model, the FP model and COnstructive COst MOdel (COCOMO) 2.0 reuse model.

A. Techniques for Maintenance

Effective software maintenance is performed using techniques specific to maintenance. The following provides some of the best practice techniques used by maintainers [1],[3].
Program Comprehension: Programmers spend considerable time in reading and comprehending programs in order to implement changes. Code browsers are a key tool in program comprehension. Clear and concise documentation can aid in program comprehension.

1) Re-engineering: Re-engineering is to examine and alter the subject system to reconstitute it in a new form. Reengineering is often not undertaken to improve maintainability but is used to replace aging legacy systems.

2) Reverse engineering: Reverse engineering is the process of analyzing a subject system to identify the system components and their interrelationships. Reverse engineering is passive, it does not change the system, or result in a new one.

3) Impact Analysis: Impact analysis identifies all systems and products affected by a change request and develops an estimate of the resources needed to accomplish the change. It

is performed after a change request enters the configuration management process.

B. Problems associated with software maintenance

Software maintenance work currently is typically much more expensive than what it should be and takes more time than required. The following discusses some of the technical and management problems relating to software evolution and maintenance[1][2].

1) Limited understanding: It is often difficult to trace the evolution of the software through it versions, changes are not documented, and the developers are usually not around to explain the code.

2) Testing: The cost of repeating full testing on a major piece of software can significant in terms of time and money.

3) Impact analysis: The software and the organization must both undergo impact analysis. Impact analysis is necessary for risk abatement.

4) Staffing: Maintenance personnel often are viewed as second class citizens and morale suffers . Maintenance is not viewed as glamorous work.

C. Software Estimation

Software project managers usually estimate the software development effort [2]. If the effort is estimated cost and duration can be easily calculated because cost is the product of working hours and wages, duration depends on how many hours a person works. Inaccurate estimations in software development industry is one of the most serious problems that cause the software projects failure. Both under and over estimations have negative impact on projects results. There are some sizing approaches for estimating the software maintenance efforts such as [3]:

1) The ACT model: ACT (Annual Change Traffic) is defined as the ratio of the system undergoing maintenance and the number of source code instructions that are modified or added during one year of maintenance.

$$ACT = \frac{KLOC_{added} + KLOC_{deleted}}{KLOC_{total}}$$

Where, $KLOC_{added}$ is the total kilo lines of source code added during maintenance. K $LOC_{deleted}$ is the total KLOC deleted during maintenance. Thus, the code that is changed, should be counted in both the code added and code deleted.
The annual change traffic(ACT) is multiplied with the total development cost to arrive at the maintenance cost, i.e.

Maintenance cost = ACT × Development cost

They are mainly applicable for annual maintenance cost estimation where an organization has the historical data for ACT.

2) Lines of code (LOC): LOC measures the size of a project by counting the number of source instructions in the developed program. While counting the number of source instructions, lines used for commenting the code and header lines are ignored. Determining the LOC count at the end of a project is very simple However, counting the LOC for a prior estimation is usually difficult.

3) The FP (Function Points) model: The FP (Function Points) model defines the five basic function types to estimate the size of the software. Two data functions types are internal logical files (ILF) and external interface files (EIF), and the remaining three transactional function types are external inputs (EI), external outputs (EO), and external inquiries (EQ).

4) Software effort prediction is needed for the following Major decision situations [2][4]:

- Making investment or other financial decisions involving a software development effort .
- Setting project budgets and schedules as a basis for planning and control .
- Deciding on or negotiating tradeoffs among software cost, schedule, functionality, performance or quality factors .
- Making software cost and schedule risk management decisions
- Deciding which parts of a software system to develop, reuse, lease, or purchase
- Making legacy software inventory decisions: what parts to modify, phase out, outsource, etc

A class of evaluation criteria that assesses the quality of future prediction is based on residual analysis and includes the PRESS (PREdiction Sum of Squares), SPR (Statistical Prediction Rule), MMRE (Mean Magnitude Relative Error) and MdMRE (Median Magnitude Relative Error) and Prediction at level r (PRED) i.e. $PRED_{25}$ and $PRED_{50}$ [5][9][26]. The resulting models constructed using neural network techniques using training and learning functions. Training and learning functions are mathematical procedures used to automatically adjust the network's weights and biases.The various functions used are: trainlm (Levenberg- Marquardt backpropagation), learngdm (Gradient descent method), tansig (Hyperbolic Tangent sigmoid) and sse (Sum squared error performance).[7]

D. Neural Networks

A neural network is good at discovering relationships and pattern in the data. Multiple layers are arranged in one succeeding the other, the first layer is called the input layer, the last layer is called the output layer, and the layers between are hidden layers. ANN trained using an algorithm learns stage wise, progressing from fairly simple to more complex mapping functions. The mean-square error decreases with an increasing number of iterations during training. Each neuron in a particular layer is connected with all neurons in the next layer. The output of a layer can be determined by equations:

$$a = x_1w_1+x_2w_2+x_3w_3...+x_nw_n$$

The connection between the i^{th} and j^{th} neuron is characterized by the weight coefficient w_{ij}. The weight coefficient reflects the degree of importance of the given connection in the neural network. The biasing is done to neutralize the network from extreme inputs, so that the network can tolerate extreme inputs. The used FFNN model is shown in Fig.1.

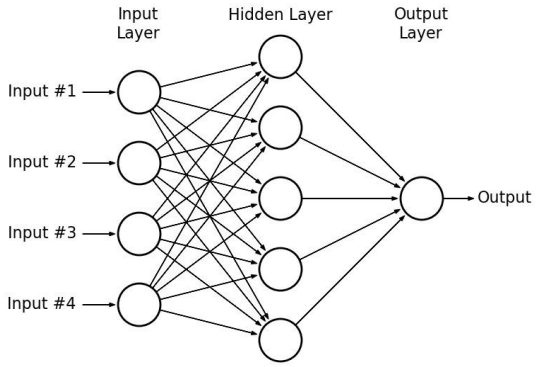

Fig.1 A Multi-Layer Feed Forward Neural Network

II. RELATED WORK

A number of research papers have been published describing the development and predict software maintenance task effort. Effort estimation is a crucial part and a key factor for successful software project planning and management. The literature shows that many studies have been conducted for effort estimation and maintenance as follows:

Martin Shepperd and Chris Schofield (1997) [8] describes an alternative approach to estimation based upon the use of analogies. The underlying principle is to characterize projects in terms of features (for example, the number of interfaces, the development method or the size of the functional requirements document). Estimation by analogy is a form of CBR. Cases are defined as abstractions of events that are limited in time and space. The key activities for estimating by analogy are the identification of a problem as a new case, the retrieval of similar cases from a repository, the reuse of knowledge derived from previous cases and the suggestion of a solution for the new case. Estimation by analogy is able to operate in circumstances where it is not possible to generate an algorithmic model.

Andrea De Lucia et al. (2002) [9] focuses on an empirical study aiming at constructing cost estimation models for corrective maintenance projects. The resulting models[5], constructed using multivariate linear regression techniques, allow to estimate the costs of a project conducted according to the adopted maintenance processes. Model performances on future observations were achieved by taking into account different corrective maintenance task typologies, each affecting the effort in a different way. A data set obtained from different corrective maintenance projects was used as case study to experimentally validate and compare model performances through multivariate linear regression models. Future work will be devoted to introduce in the maintenance projects different metric plans aiming at characterizing the difference in the maintenance projects of the subject organization.

Yunsik Ahn et al. (2003) [10] has proposed the software maintenance project effort estimation model (SMPEEM)[6] is which is based on the function point measure and new maintenance productivity factors. Finally, Yunsik Ahn et al. suggest an exponential function model which can show the relationships among the maintenance efforts, maintenance environment factors, and function points of the software maintenance project. Approaches for estimating the software maintenance efforts such as source lines of code (SLOC), function points (FPs) , and object points. Yunsik Ahn et al. review the annual change traffic (ACT) model, the FP model and COnstruction COst MOdel (COCOMO) 2.0 reuse model. For SMPEEM the collected data are only from small maintenance projects. The collected data are only from small maintenance projects.

Harry M. Sneed & Peter Brössler (2003) [11] identify those factors, which are critical to the success of a maintenance operation in general and to apply them to a particular maintenance project. Eight factors are defined and evaluated in accordance with the existing literature on software maintenance and with the experience gained on several such maintenance projects. The factors are: functionality, quality, complexity, volatility, Costs, Release deadlines, User satisfaction and Profitability. The study is based on empirical data collected over the duration of the project and is intended to contribute to the overall knowledge of software maintenance management. Research is still out on the last two factors – user satisfaction and profitability. However, preservation of the existing functionality, quality, complexity, volatility, productivity and punctuality – the maintenance project can be considered to be relatively successful.

Jaswinder Kaur et. al (2008) [12] done comparative analysis among existing Halstead Model, Walston-Felix Model, Bailey-Basili Model, Doty Model and Neural Network Based Model is performed. Neural Network has outperformed the other considered models. They proposed Neural Network system as a soft computing approach to model the effort estimation of the software systems. As Neural based system is able to approximate the nonlinear function with more precision and non of the researcher have explored Neuro approach for the Effort Estimation and there is still scope of exploring more statistical modeling approaches. Perform the comparison of the models on basis of Mean Magnitude of Relative Error (MMRE) & Root Mean Square Error (RMSSE). The dataset of NASA is used for the comparison of different models. They suggested to use of Neuro based technique to build suitable generalized type of model that can be used for the software effort estimation of all types of the projects.

Shahid Hussain et al. (2009) [13] proposed a RCM (Reduce Corrective Maintenance) model which represents the implementation process model which comprises on filling and analyzing process of checklists in each phase. The corrective maintenance efforts are increases due to flaws remains in other phase of software development life cycle. These flaws can be overcome if stakeholders fully understand the activities of each concern phase. Authors proposed a RCM model which

comprises on filling and analyzing process of checklists in each phase. If all stakeholders of each phase filled the checklist in precise manner then evaluated result of each checklist shows that how much stakeholder have understand the activities. The RCM (Reduce Corrective Maintenance) model[8] comprises on filling and analyzing process of checklists in each phase.

Ch. Satyananda Reddy and KVSVN Raju (2009) [14] constructed a cost estimation model based on artificial neural networks. The model is designed to improve the performance of the network that suits to the COCOMO model.The feed forward multilayer perceptron with back propagation learning algorithm are the most commonly used in the cost estimation field. The network is trained with back propagation learning algorithm by iteratively processing a set of training samples and comparing the network's prediction with the actual effort. The technique in the use of the neural network for predicting software cost estimation is back propagation trained multilayered feed forward networks with sigmoidal activation function. COCOMO dataset is used to train and to test the network and observed that proposed neural network model improves the estimation accuracy of the model. This work can be extended by integrating this approach with fuzzy logic to effectively deal with imprecise and uncertain information associated with COCOMO.

Parvinder S. Sandhu et al. (2009) [15] experimented Statistical Models, Fuzzy-GA and Neuro-Fuzzy (NF) Inference Systems to estimate the software effort for projects. The performances of the developed models were tested on NASA software project datasets and results are compared with the Halstead, Walston-Felix, Bailey-Basili, Doty and Genetic Algorithm Based models mentioned in the literature. On comparison, the results shows that the NeurofuzzyModel has the lowest MMRE and RMSE values . Neuro-fuzzy Model shows the better results as compared with the Fuzzy-GA based hybrid Inference System, Linear Statistical Models and Pure Quadratic Statistical Model for the Effort Prediction. Hence, the developed Neuro-Fuzzy model is able to provide good estimation capabilities. It is suggested to use of Neuro-Fuzzy technique to build suitable model structure for the software effort.

Ch. Satyananda Reddy and KVSVN Raju (2010) [16] constructs software effort estimation model based on artificial neural networks.It is proposed to use single layer feed forward neural network to accommodate the model and its parameters to estimate software development effort. The network is trained with back propagation learning algorithm and Resilient Back propagation algorithm (RPROP) by iteratively processing a set of training samples and comparing the network's prediction with the actual effort. COCOMO dataset is used to train and to test the network and it was observed that proposed neural network model improves the estimation accuracy of the model. The test results from the trained neural network are compared with that of the COCOMO model. By comparing the results of these two models, it is proven that both models (SLANN with BP and SLANN with RPROP) works better than COCOMO and SLANN with RPROP is an optimal neural network model for software effort estimation. SLANN with BP works well only for projects with small size, where as SLANN with RPROP works well for all kinds of projects as the convergence rate of RPROP algorithm is very high.

Ruchi Shukla and A K Misra (2010) [17] presents a neural network (NN) approach to model and predict the software maintenance effort based on an available real life dataset of outsourced maintenance projects. A comparison between results obtained by NN and regression modeling is also presented. It is concluded that NN is able to successfully model the complex, non-linear relationship between a large number of effort drivers and the software maintenance effort, with results closely matching the effort estimated by experts. The NN model trained using experimental data was found to have good generalization capabilities and is able to successfully predict the effort closely matching the experimental observations. Since the effect of various cost drivers on effort is often quite complex, ANN can be used as an effective tool to model and predict the SM effort. However, the models should also be evaluated by exploring the model sensitivity and scalability on a variety of historical and unseen input data.

Dr. Arvinder Kaur et al. (2010) [18] evaluate and compare the application of different soft computing techniques – Artificial Neural Networks, Fuzzy Inference Systems and Adaptive Neuro-Fuzzy Inference Systems to construct models for prediction of Software Maintenance Effort. The maintenance effort data of two commercial software products is used in this study. The dependent variable in the study is maintenance effort. The independent variables are eight Object Oriented metrics . It is observed that soft computing techniques can be used for constructing accurate models for prediction of software maintenance effort and Adaptive Neuro Fuzzy Inference System technique gives the most accurate model. it concludes that soft computing techniques can be successfully used for prediction of software maintenance effort.Results need to be generalized by conducting similar studies on maintenance data of other software systems.

Ali Bou Nassif et al. (2011) [19] presents a novel regression model to estimate the software effort based on the use case point size metric. The use case point model takes use case diagrams as input and gives the software size in use case points as output. The proposed effort equation takes into consideration the non-linear relationship between software size and software effort, as well as the influences of project complexity and productivity presents a novel regression model to estimate the software effort based on the use case point size metric. The use case point model takes use case diagrams as input and gives the software size in use case points as output. The proposed effort equation takes into consideration the non-linear relationship between software size and software effort, as well as the influences of project complexity and productivity.

Mr. Sandeep Sharawat(2012) [20] focus on MI (Maintainability Index) which is a composite metric that incorporates a number of traditional source code metrics into a single number that indicates relative maintainability. Li-Henry data is used for the prediction of MI to train neural networks. After training the neural network, a set of input values are provided & change , MI can be determined as output of the trained neural network. further design own training algorithm for better results focusing on MI, those may be based on

Fuzzy Logic algorithms, Artificial Intelligence based algorithms or Support vector based algorithms etc. While designing own algorithms focusing on MI will result in more accuracy in the prediction of MI to show how maintainable software system will be, which may have major impact in industries as companies may be able to predict MI, so that risk analysis or cost analysis can be considered earlier in respect of software maintenance, which will help to reduce overall cost of software systems.

Petr Marounek (2012/13) [21] introduces the area of software support and maintenance -effort estimation approaches. Software maintenance is a set of activities needed for cost-effective support of IT solution. Petr Marounek introduces simplified, easy to use approach to effort estimation in software maintenance based on extending PERT formula about quality of estimator and historical experience. Both formulas were verified in sub-competence center for supporting mortgage IS with significantly better result than only pure PERT estimate (98.8% and 91.8% against pure PERT 90.1%) and aware of short historical row of values, which might affect calculated results and therefore recommends validation on longer row of values in the future. There are additional opened issues to be solved in further research resulting from author's concept of dividing competence centre into two parts - Competence center for support and maintenance, and Sub-competence center for support and maintenance of selected solution (or related solutions).

E.Praynlin and P.Latha (2013) [22] uses Adaptive Neuro fuzzy Inference system (ANFIS) model for software effort estimation. The advantage of both the neural network and fuzzy logic can be combined by using the neuro fuzzy model. ANFIS used in this paper is of sugeno type Fuzzy inference system. It applies the combination of Least square method and the back propagation gradient descent method. The functioning of ANFIS is a five-layered feed-forward neural structure and various type of membership functions are used. ANFIS Uses two set of datasets. One set of dataset consists of 63 projects and other consists of 93 datasets both are from the historic projects of NASA. ANFIS generates FIS by two main methods i.e. Grid partitioning and Clustering. The objective of E.Praynlin and P.Latha is to provide which membership function is to be used in ANFIS which gives accurate predictive effort. It was concluded that the proposed ANFIS model using Trapezoidal membership function has low MMRE than the above mentioned membership functions.

Jyoti G. Borade and Vikas R. Khalkar (2013) [23] describes several existing methods for software project effort and cost estimation[10].Software cost estimation is the process of predicting the amount effort required to build a software system. Estimation is a complex activity that requires knowledge of a number of key attributes. At the initial stage of a project, there is high uncertainty about these project attributes. Conventional estimation techniques focus only on the actual development effort furthermore, this paper also described test effort estimation. Testing activities make up 40% total software development effort. Jyoti G. Borade [10] in Software Project Effort and Cost Estimation Techniques states that no model can estimate the cost of software with high degree of accuracy. There are various techniques used in software cost estimation. By following Boehm's classification system, these Methods are summarize into three categories expert judgment, algorithmic estimation, and analogy based estimation.

Olga Fedotova et al. (2013) [24] presents common methods used in the software effort estimation (SEE) and the study performed in a software development organization (SDO) that is implementing the software development process improvement framework Capability Maturity Model Integrated(CMMI). The aim of this paper is to present the main software effort estimation methods, their advantages and disadvantages and to apply a formal method based on the Multiple Linear Regression technique to the historical data of a medium-sized multinational software company. The stepwise Multiple Linear Regression (MLR) technique was selected and used for the software development and software testing processes. The results achieved with MLR were compared with the estimates provided by the area expert. The model obtained for the testing team performed better results than the expert judgments.

Sonika Basra et al. [25] presents a software effort estimation tool based on artificial neural networks.The test results from the trained neural network are compared to the COCOMO II model and also to the neural network trained using default Levenberg-Marquardt optimization by computing the Magnitude of Relative Error (MRE) values are calculated for each project. The COCOMO II model mapped to an ANN with two hidden layers and ten nodes in each layer to increase the accuracy of the network. The neural network used to predict the software development effort is the multilayer feed forward neural network with the identity function at input, hidden and output units. PROMISE dataset was used to train and to test the network and it was observed that our ANN model provides better cost estimations than the estimations produced by the COCOMO II model. This work can be extended by integrating this approach with fuzzy logic to effectively deal with imprecise and uncertain information associated with COCOMO II.

III. CONCLUSION

The current research is involved in several legacy system migration and web-based integration projects and it is likely that the software systems being maintained in the future will be based on heterogeneous technological platforms thus requiring different skills in the maintenance teams. Yunsik Ahn[10] proposed in SMPEEM that the collected data are only from small maintenance projects.The result is that the methodology of SMPEEM[6] may be used as a useful guide for the estimation of a maintenance project in any organization. After that eight factors[26] are defined and evaluated in accordance with the existing literature on software maintenance and with the experience gained on several such maintenance projects. The RCM (Reduce Corrective Maintenance) [13] model comprises on filling and analyzing process of checklists in each phase. Moreover, the checklist of RCM can be updated by stakeholder who will apply this model during development process of software. Jyoti G. Borade[23] in Software Project Effort and Cost Estimation Techniques states that no model can estimate the cost of software with high degree of accuracy. Estimation is a complex activity that requires knowledge of a number of key attributes. No model can estimate the cost of software

with high degree of accuracy. Estimation is a complex activity that requires knowledge of a number of key attributes. Neural Network system is used as a soft computing approach to model the effort estimation of the software systems. Multi layer feed forward neural network used to accommodate the model and its parameters to estimate software development effort [14]. Parvinder S. Sandhu [15] experimented Statistical Models, Fuzzy-GA and Neuro-Fuzzy (NF) Inference Systems to estimate the software effort for projects. Later, MI (Maintainability Index) used which is a composite metric that incorporates a number of traditional source code metrics into a single number that indicates relative maintainability. Petr Marounek[21] introduces simplified, easy to use approach to effort estimation in software maintenance based on extending PERT formula about quality of estimator and historical experience.

REFERENCES

[1] Thomas M. Pigoski, Technical Software Services (TECHSOFT), Inc..Software Maintenance. IEEE – Trial Version 1.00 – May 2001.

[2] R.S. Pressman. Software Engineering: A Practitioner's Approach. McGraw-Hill, fifth edition,2001.

[3] Rajib Mall. Software Engineering: Fundamentals of Software Engineering. PHI, third edition,2009.

[4] M M Lehman,Laws of Software Evolution Revisited, 21/1/97, 5:20 pm, mm l556/2 papers.

[5] Katrina D. Maxwell. Applied Statistics for Software Mangers. Pearson Education 2002 edition.

[6] J. W. Bailey and V. R. Basili, "A meta model for software development resource expenditure," in Proceedings of the International Conference on Software Engineering, pp. 107–115, 1981.

[7] Neural network specification, available at URL: http://www.mathworks.in/products/neural-network/description3.html.

[8] Martin Shepperd and Chris Schofield. Estimating Software Project Effort Using Analogies.IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 23, NO. 12, NOVEMBER 1997.

[9] Andrea De Lucia, Eugenio Pompella and Silvio Stefanucci. Effort Estimation for Corrective Software Maintenance. SEKE '02, July 15-19, 2002.

[10] Yunsik Ahn, Jungseok Suh, Seungryeol Kim and Hyunsoo Kim. The software maintenance project effort estimation model based on function points. 2003; 15:71–85.

[11] Harry M. Sneed & Peter Brössler. Critical Success Factors in Software Maintenance-A Case Study. Proceedings of the International Conference on Software Maintenance (ICSM'03),2003 IEEE.

[12] Jaswinder Kaur, Satwinder Singh, and Karanjeet Singh Kahlon. Comparative Analysis of the Software Effort Estimation Models. World Academy of Science, Engineering and Technology 46 2008.

[13] Shahid Hussain, Muhammad Zubair Asghar, Bashir Ahmad and Shakeel Ahmad. A Step towards Software Corrective Maintenance: Using RCM model.(IJCSIS) International Journal of Computer Science and Information Security Vol. 4, No. 1 & 2, 2009.

[14] Ch. Satyananda Reddy and KVSVN Raju. A Concise Neural Network Model for Estimating Software Effort. International Journal of Recent Trends in Engineering, Issue. 1, Vol. 1, May 2009.

[15] Parvinder S. Sandhu, Manisha Prashar, Pourush Bassi, and Atul Bisht. A Model for Estimation of Efforts in Development of Software Systems.World Academy of Science, Engineering and Technology Vol:3 , 26, August 2009.

[16] Ch. Satyananda Reddy and KVSVN Raju. An Optimal Neural Network Model for Software Effort Estimation. Int.J. of Software Engineering, IJSE Vol.3 No.1 January 2010.

[17] Ruchi Shukla and A K Misra. Software Maintenance Effort Estimation – Neural Network Vs Regression Modeling Approach, International Journal of Computer Applications (0975 - 8887) Volume 1 – No. 29, 2010.

[18] Dr. Arvinder Kaur, Kamaldeep Kaur and Dr. Ruchika Malhotra. Soft Computing Approaches for Prediction of Software Maintenance Effort**,** International Journal of Computer Applications (0975 - 8887) Volume 1 – No. 16, 2010.

[19] Ali Bou Nassif, Danny Ho and Luiz Fernando Capretz. Regression Model for Software Effort Estimation Based on the Use Case Point Method. International Conference on Computer and Software Modeling IPCSIT vol.14 (2011).

[20] Sandeep Sharawat. Software Maintainability Prediction Using Neural Networks, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, Vol. 2, Issue 2, pp.750-755, Mar-Apr 2012.

[21] Petr Marounek. Simplified approach to effort estimation in software maintenance. Journal of Systems Integration 2012/3.

[22] E.Praynlin and P.Latha. Estimating Development Effort of Software Projects using ANFIS. ICON3C 2012 , IJCA.

[23] Jyoti G. Borade and Vikas R. Khalkar. Software Project Effort and Cost Estimation Techniques**,** IJARCSSE, Volume 3, Issue 8, August 2013.

[24] Olga Fedotova, Leonor Teixeira and Helena Alvelos Software Effort Estimation with Multiple Linear Regression: Review and Practical Application. Journal of Information Science And Engineering 29, 925-945 (2013).

[25] Sonika Basra, Nitin Bhatia, and Vijay Kumar Mago. SoftEE: A Tool for Software Effort Estimation using Back Propagation Neural Network and Bayesian Regularization.

[26] Magne Jørgensen. A Critique of How We Measure and Interpret the Accuracy of Software Development Effort Estimation.