www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 6 June 2017, Page No. 21808-21813
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i6.42

Artificial Neural Network Model for predicting Maintainability Using MOOD and size metrics

Rajbinder Kaur[1], Mehak Aggarwal[2]
[1]Research Scholar, [2]Associate Professor
[1,2]Department of Computer Science and Engineering, Lala Lajpat Rai Institute of Engineering & Technology, Moga, (Punjab) INDIA
[1]brar.rajbinder@gmail.com, [2]aggarwal.mehak@gmail.com

***Abstract*—One of the major challenges to software industry today is to provide products with high degrees of quality and functionality. Maintainability is one such quality attribute that accounts for 40-70% of the total cost of the project. As the technology advances, a number of metrics such as CK (Chidamer & Kemerer), MOOD (Metrics for Object Oriented Design), Lorenz and Kidd Suite, etc have been proposed to predict quality characteristics such as maintainability, reliability, usability etc. Currently, software development is mostly based on object oriented paradigm. At the system level, there are patterns that represent the extent of use of encapsulation, inheritance, polymorphism or cooperation among classes which are closely related with the quality characteristics. By finding those patterns developer of a project can say that a certain design is more maintainable than another. Most of the Maintainability models proposed earlier are based on CK metrics. CK are class based metrics but MOOD metrics are project based and represents all the basic mechanisms of Object Oriented Paradigm. Size of project also plays a significant role in maintainability prediction. In particular, larger size systems are hard to analyze and understand. Earlier statistical models were proposed but nowadays machine learning techniques such as Artificial Neural Networks (ANN), fuzzy, neurofuzzy etc. are used. ANN is capable of modeling complex functions and has strong generalization ability. Hence, in this paper MOOD and size metric (Lines of Code) based ANN model is proposed to predict maintainability of a project early in the design phase.**

Keywords—Artificial Neural Networks; MOOD metrics; Software Quality; Principal Component Analysis; Size metrics.

INTRODUCTION

Currently, software development is mostly desired to be based on Object Oriented paradigm and software quality is a major factor of concern. The quality of Object Oriented software can be best assessed by the use of software metrics. Even today the essential of software quality engineering is to investigate the relationships among the software metrics and end-product quality, and, based on the findings, to engineer improvements in both process and product quality. Maintainability is an important quality attribute and a difficult concept as it involves a number of measurements. Despite the fact that software maintenance is an expensive and challenging task, it is not properly managed and often ignored. One reason for this poor management is the lack of proven measures for software maintainability. The backbone of any software system is its design. It is the skeleton where the flesh i. e. code, will be supported. Measuring software maintainability early in the development life cycle, especially at the design phase, may help designers to incorporate required enhancement and corrections for improving maintainability of the final software.

Artificial Neural Networks can be used as a predictive model because it is very sophisticated modeling techniques capable of modeling complex functions. Unlike mathematical models that require precise knowledge of all contributing variable, a trained artificial neural network can estimate process behavior. It is proven fact that neural nets have a strong generalization ability, which means that, once they have been properly trained, they are able to provide accurate results even for cases they have never seen before.

In this paper, ANN based prediction model to predict maintainability of object oriented systems at design phase has been proposed and validated against the empirical results provided by Abreu et al.[3]

The outline of this paper is the following: section 2 presents the overview of the existing studies; section 3 describes the proposed model ; validation of the proposed model and results are given in section 4 and finally section 5 concludes the paper and provides future scope for the research.

RELATED WORK

Several maintainability models/methodologies were proposed to help the designers in calculating the maintainability of software so as to develop better and improved software systems. Many researchers have worked to find quality of object oriented systems and hence several metrics were proposed. Fernando Brito e Abreu et al. (1996)[3], reviewed a set of metrics called MOOD, suited for evaluating the use of object oriented mechanisms. They conducted experiment on eight small sized information management systems. The impact of Object-Oriented design on software quality characteristics is evaluated. Data obtained in this experiment show how OO design mechanisms such as inheritance, polymorphism, information hiding and coupling, can influence quality characteristics like reliability or maintainability. Quah and Thwin (2002)[15] in their research envisaged the quantity of faults in a particular class by employing a multiple regression model and a neural network model. Three industrial real-time subsystems data were implicated in their study and it was found that neural network model has more accurate prediction than regression model. Muktamyee Sarker(2005)[14], suggest that, only those metrics should be used which are empirically validated. K. K. Aggarwal et al.(2008)[6] examined the application of ANN for software quality prediction using Object- Oriented (OO) metrics. The results showed that the Mean Absolute Relative Error (MARE) was 0.265 of ANN model. Thus ANN method was useful in constructing software quality model. Sanjay

Kumar Dubey et al.(2012)[7] predicted the maintainability of object oriented software by using neural network model. They used UIMS dataset and CK metrics. The comprehensive experimental analysis showed that the proposed model is quite valuable for real life applications. Alisara Hincheeranan and Wanchai Rivepiboon(2012)[5] present a multivariate linear regression to establish the maintainability estimation model. They considered flexibility and extendibility as two sub characteristics of maintainability. Gurpreet Kaur and Mehak Aggarwal (2014)[9] proposed a Fuzzy Model for Evaluating the Maintainability of Object Oriented System Using MOOD Metrics. Rajiv D. Bankar et all. [16] examines the relationship between software maintainability and complexity. They showed that complexity is inevitably dependent on size of project. Excluding size and complexity measures makes the model for maintainability prediction incomplete. Thus size metrics such as lines of code are important.

METRICS THAT AFFECT MAINTAINABILITY

A. Metrics Studied

In this paper, focus is on MOOD and size metrics. MOOD (Metrics for Object Oriented Design)[3] includes the following metrics:

Attribute Inheritance Factor (AIF):

The AIF numerator is the sum of inherited attributes in all classes of the system under consideration. The AIF denominator is the total number of available attributes (locally defined plus inherited) for all classes.

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)} \quad (1)$$

TC= Total Number of Classes.

A_i =Number of Inherited attributes of Class C_i.

A_a =Number of Total attributes of Class C_i.

Method Inheritance Factor (MIF):

The MIF numerator is the sum of inherited methods in all classes of the system under consideration. The MIF denominator is the total number of available methods (locally defined plus inherited) for all classes.

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)} \quad (2)$$

M_i =Number of Inherited methods of Class C_i.

M_a =Number of Total methods available in Class C_i.

Attribute Hiding Factor (AHF):

The AHF numerator is the sum of the invisibilities of all attributes defined in all classes. The invisibility of an attribute is the percentage of the total classes from which this attribute is not visible.

$$AHF = \frac{\sum_{i=1}^{TC} A_h(C_i)}{\sum_{i=1}^{TC} A_d(C_i)} \quad (3)$$

A_h =Number of hidden attributes of Class C_i.

A_d=Number of total attributes (available and hidden).

Method Hiding Factor (MHF):

The MHF numerator is the sum of the invisibilities of all methods defined in all classes. The invisibility of a method is the percentage of the total classes from which this method is not visible.

$$MHF = \frac{\sum_{i=1}^{TC} M_h(C_i)}{\sum_{i=1}^{TC} M_d(C_i)} \quad (4)$$

M_h =Number of Inherited methods of Class C_i.

M_d=Number of Total methods of Class C_i.

Polymorphism Factor (PF):

The POF numerator represents the actual number of possible different polymorphic situations. The POF denominator represents the maximum number of possible distinct polymorphic situations for class Ci. This would be the case where all new methods defined in Ci would be overridden in all of their derived classes.

$$POF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]} \quad (5)$$

M_o =Number of methods overridden of Class C_i.

M_n =Number of new methods.

DC=Descendants count

Coupling Factor (CF):

The COF denominator stands for the maximum possible number of couplings in a system with TC classes. The client supplier relation (represented by Cc =>Cs) means that Cc (client class) contains at least one non inheritance reference to a feature (method or attribute) of class Cs (supplier class). The COF numerator then represents the actual number of couplings not imputable to inheritance.

$$COF = \frac{\sum_{i=1}^{TC} [\sum_{j=1}^{TC} is\_client(C_i, C_j)]}{TC^2 - TC} \quad (6)$$

Where

$$is\_client(C_i, C_j) = \begin{cases} 1 & C_i \Rightarrow C_j \wedge C_i \neq C_j \\ 0 & otherwise \end{cases}$$

Each of these metrics refer to a basic structural mechanism of object oriented paradigm as encapsulation (MHF and AHF), inheritance (MIF and AIF), polymorphism (PF) and message-passing (CF).

Size Metric

Source lines of code (SLOC), also known as lines of code(LOC), is a software metric used to measure the size of a computer program by counting the number of lines in the text of the program's source code. SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or maintainability once the software is produced. It is fairly intuitive that the total size of a system should feature heavily in any measure of maintainability. A larger system requires, in general, a larger effort to maintain.

Correlation of Size and Rework

A correlation coefficient is a statistical measure of the degree to which changes to the value of one variable predict change to the value of another. **Size** is considerable factor in measuring the maintainability of a software, which is shown by correlation coefficient results as shown:

CORRELATION COEFFICIENT

	Size
Rework	0.8885

DOI: 10.18535/ijecs/v6i6.42

PROPOSED ANN MODEL

B. Proposed Model

The goal of our study is to explore the relationship between OO metrics and maintenance effort at the project level.

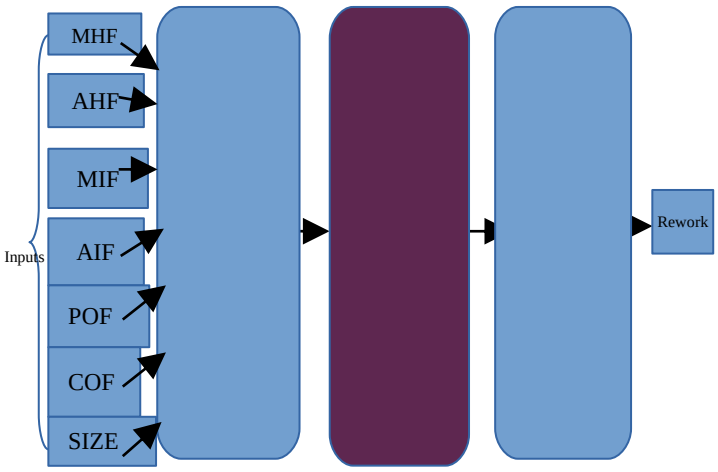


Figure: Proposed Model

Maintainability is taken as number of hours spent to make required changes in the software i. e. rework effort.

Rework = f (MIF, AIF, MHF, AHF, CF, POF, size)

Equation...

Maintainability is inversely proportion to Rework i.e. efforts required to maintain software.

- Input Unit consists of all the independent variables of this model. Independent variables are those that can characterize the project and influence the evaluation results. Data is gathered from research papers. (Abreu, 1996), (Harrison, 1998), (Abreu, 1996). Data is collected empirically by the authors using MOODKIT tool. The rework is measured by using the number of hours spent to correct bugs and make required changes in a project. A line change could be an addition or a deletion.
- PREPROCESSING UNIT: Nowadays, in the ANN, the high computational cost connected with the learning process of the MLP (Multi-Layer Perceptron Network) is a grave problem. This cost is directly linked with the training dataset size. The computational cost of ANN training can be reduced by introducing preprocessing techniques such as; Min-Max, Z-Score and Decimal Scaling Normalization. The preprocessing techniques increase the robustness of the proposed model and increase the training efficacy of MLP's. One of the first steps concerns the normalization of the data. This step is very important when dealing with parameters of different units and scales. All parameters should have the same scale for a fair comparison between them. The standard score of a raw data is calculated by the formula:

$$z = x - \frac{x}{\sigma}$$

Where:

μ *is* the mean of the population.

σ *is* the standard deviation of the population.

It returns the z-score for each element of 'x' such that columns of 'x' are centered to have mean 0 and scaled to have standard deviation 1.

The performance of a MLP very much depends on its generalization capability, which in turn is dependent upon the data representation. One important characteristic of data representation is uncorrelated. In other words, a set of data presented to a MLP ought not consist of correlation information. This is because correlated data reduce the distinctiveness of data representation and thus, introduce confusion to the MLP model during the learning process and hence, producing one that has low generalization capability to resolve unseen data. This suggests a need for eliminating correlation in the sample data before they are being presented to an MLP. This can be achieved by applying the **Principal Component Analysis (PCA)** technique onto input data sets prior to the MLP training process as well as interpretation stage. Steps involved in Principal Component Analysis are:

1. Most often, the first step in PCA is to standardize the data.
2. Calculating the coefficients of the principal components and their respective variances is done by finding the eigen functions of the sample covariance matrix:

   [V D] = eig(cov(z))

   The matrix V contains the coefficients for the principal components. The diagonal elements of D store the variance of the respective principal components.
3. To calculate the principal components simply multiply the standardized data by the principal component Coefficients.

   Principal Components = B * V

TABLE I. PRINCIPAL COMPONENTS

	PC1	PC2	PC3
MHF	0.2549	-2.2617	0.1439
AHF	-0.6949	-0.00024	**-1.1001**
MIF	-0.5426	2.6399	-1.5256
AIF	0.2255	**-1.3384**	0.7997
POF	1.0956	-0.1352	-0.4318
COF	-1.0473	0.3284	0.5083
Size	-0.3792	**-0.7127**	-0.4402
Eigen value	2.8096	1.2398	1.1220
Commulative Variance	40%	57.8%	73.88%

- ARTIFICIAL NEURAL NETWORK UNIT: Artificial neural networks (ANNs) are a family of models inspired by the biological neural networks. ANNs are used to estimate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected neurons which exchange messages between each other. The connections have numeric weights that can be tuned

based on experience, making neural nets adaptive to inputs and capable of learning.

Architecture of Model selected is shown in Table II. Here, MLP neural network with three layers is used. Input layer consist of seven input neurons, hidden layer consist of ten neurons and one neuron at output layer. Data Splitting is an important step in the artificial neural network (ANN) development process whereby data is divided into training, test and validation subsets to ensure good generalization ability of the model. In the present work, dividerand() function is used for data splitting. For input layer, linear activation function is used i. e., the output of the input layer is treated as input of the input layer. For hidden layer, sigmoidal function or squashed-S function is used. For output layer linear activation function is used. Sigmoid functions are often used in artificial neural networks to introduce nonlinearity in the model.

Trainlm is used as training algorithm. Levenberg–Marquardt backpropagation (trainlm) algorithm locates the minimum of a multivariate function that can be expressed as the sum of squares of non-linear real-valued functions. It is an iterative technique that works in such a way that performance function will always be reduced in each iteration of the algorithm.

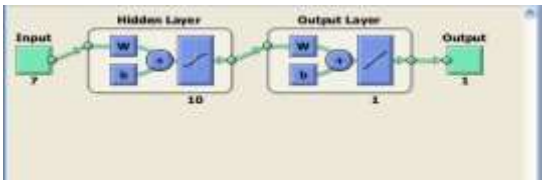


TABLE II. ANN DETAILS:

Architecure	
Number of Layers	3
Number of Hidden Neurons	10
Number of Input Neurons	7
Number of Output Units	1
Training	
Training Function	TRAINLM
Performance Function	MSE
Network Type	Feedforward
Algorithm	Back Propagation

a.

Mean Square Error is used to access the quality of the predictor. It measures the network's performance according to the mean of squared errors.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (Y_i - X_i)^2$$

where **Y** is vector of **n** predictions and **X** is vector of observed values.

MSE is used as training stopping criteria. Weights of the neural network are adjusted to keep mean square error to minimum in training data.

- After the Artificial Neural is trained, it gives results in Normalized form. As the outputs of ANN are not in real scale, post processing is required. This is done by reversing the z-score function. The value of rework is calculated by the formula:

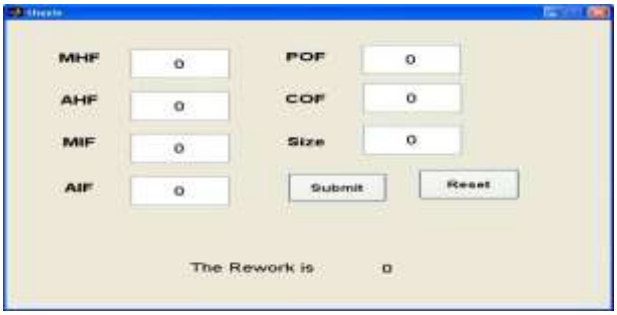
$$x = \sigma * z + \mu$$

Where:

μ *is* the mean of the population.

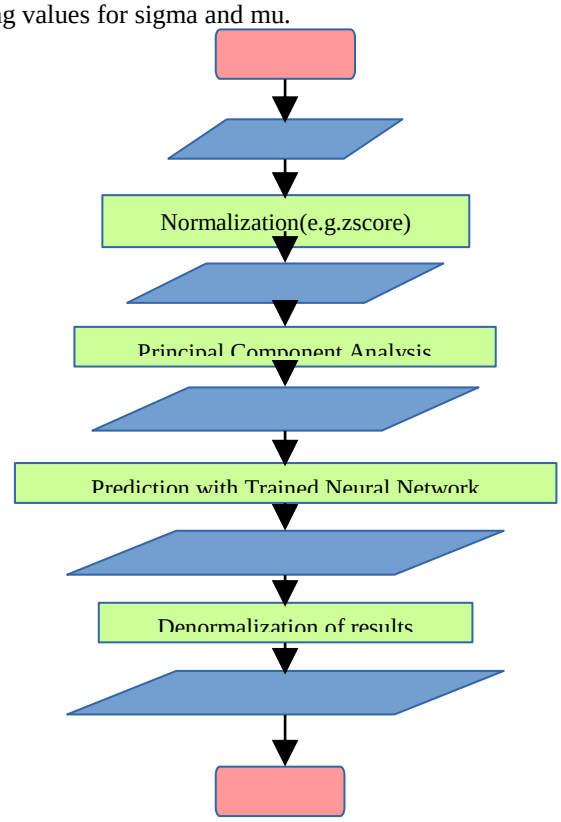
σ *is* the standard deviation of the population.

- The results should be displayed in easy to use form. Thus, Graphical User Interface is designed, where user can input values of MOOD Metrics and SIZE metrics.



WORKING OF THE PROPOSED MODEL

The working of proposed model is shown in Figure.... User input the Raw Data for all the MOOD metric variables. MOOD values are in form of percent. Size in Lines of code is entered. Input raw data is Normalized by zscore values for sigma σ and mu μ. μ is the mean of the population and σ *is* the standard deviation of the population as calculated during ANN modeling. Then normalized data is converted into Principal Coefficients. Normalized data is multiplied with coefficient values produced during ANN modeling. Principal Components are given as input to the trained ANN and then Output for the rework is produced in normalized form. Finally, results are denormalized by reversing the zcsore process and using values for sigma and mu.



DOI: 10.18535/ijecs/v6i6.42

RESULTS AND VALIDATION

ANN Results

The trained neural network **performance** progress is shown below. It shows validation and testing curves are very similar. Thus training proceeded correctly.

Fig. 1. *Performance progress of ANN*

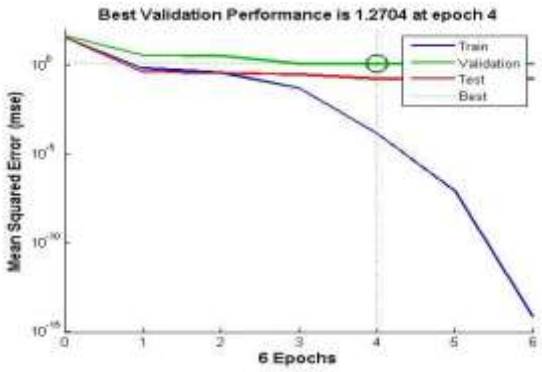
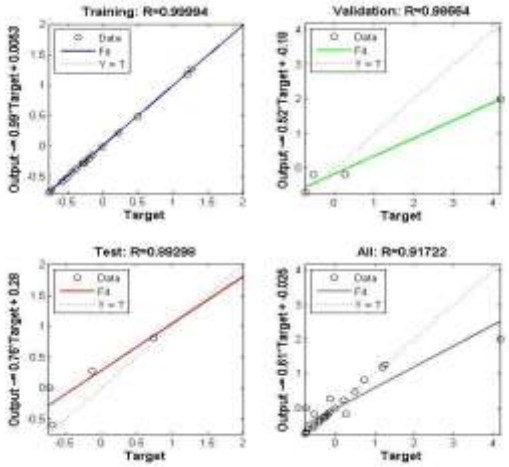


Fig. 2. Regression Plot of Trained ANN



The **Regression plot** shows the relationship between outputs and targets of network. If R = 1, this indicates that there is an exact linear relationship between outputs and targets. If R is close to zero, then there is no linear relationship between outputs and targets. The three plots represent the training, validation, and testing data. The dashed line in each plot represents the perfect result i. e. outputs = targets. Thus, this Neural Network is **validated**, since R>0.9.

Validation Of the Proposed Model

The model is validated on the basis of Response Coefficients. According to Fernando Brito e Abreu et all. [3], the response coefficients should be

RESPONSE COEFFICIENTS OF MOOD WITH REWORK

MHF	AHF	MIF	AIF	POF	COF
23.77	-1.57	5.66	-16.94	-57.93	62.07

Response of Rework with increase in value of MHF, while all other attributes are constant is shown in TABLE V. It is positive and thus valid. Similarly, Response of Rework with increase in value of AHF is shown in TABLE VI. It is negative i. e. decreases with increase in AHF as desired. Response of Rework with increase in value of MIF is shown in TABLE VI. It is positive as desired. Response of Rework with increase in value of AIF is shown in TABLE VII.

RESPONSE OF REWORK WITH INCREASE IN MHF

MHF	AHF	MIF	AIF	POF	COF	SIZE	REWORK
5	60	50	50	10	10	20000	-4.16
10	60	50	50	10	10	20000	29.88
15	60	50	50	10	10	20000	64.19
20	60	50	50	10	10	20000	95.98
25	60	50	50	10	10	20000	123.73
30	60	50	50	10	10	20000	147.02
35	60	50	50	10	10	20000	166.12
40	60	50	50	10	10	20000	181.63
45	60	50	50	10	10	20000	194.35
50	60	50	50	10	10	20000	205.12

RESPONSE OF REWORK WITH INCREASE IN AHF

MHF	AHF	MIF	AIF	POF	COF	SIZE	REWORK
20	50	50	50	10	10	20000	111.71
20	54	50	50	10	10	20000	106.96
20	58	50	50	10	10	20000	100.10
20	60	50	50	10	10	20000	95.98
20	64	50	50	10	10	20000	86.78
20	68	50	50	10	10	20000	76.90
20	70	50	50	10	10	20000	71.94
20	75	50	50	10	10	20000	60.28
20	80	50	50	10	10	20000	50.55
20	85	50	50	10	10	20000	43.41

RESPONSE OF REWORK WITH INCREASE IN MIF

MHF	AHF	MIF	AIF	POF	COF	SIZE	REWORK
20	60	5	50	10	10	20000	25.75
20	60	10	50	10	10	20000	34.54
20	60	15	50	10	10	20000	43.27
20	60	20	50	10	10	20000	51.87
20	60	25	50	10	10	20000	60.22
20	60	30	50	10	10	20000	68.25
20	60	35	50	10	10	20000	75.88
20	60	40	50	10	10	20000	83.06
20	60	45	50	10	10	20000	89.76
20	60	50	50	10	10	20000	95.98

RESPONSE OF REWORK WITH INCREASE IN AIF

MHF	AHF	MIF	AIF	POF	COF	SIZE	REWORK
20	60	50	5	10	10	20000	169.37
20	60	50	10	10	10	20000	168.58
20	60	50	15	10	10	20000	167.05
20	60	50	20	10	10	20000	164.01
20	60	50	25	10	10	20000	158.71
20	60	50	30	10	10	20000	150.68
20	60	50	35	10	10	20000	139.96
20	60	50	40	10	10	20000	126.97
20	60	50	45	10	10	20000	112.19
20	60	50	50	10	10	20000	95.98

Response is negative as desired.

TABLE IX shows Response of REWORK with increase

in POF, which is negative as desired. Response of REWORK with increase in COF is shown by TABLE X.

RESPONSE OF REWORK WITH INCREASE IN POF

MHF	AHF	MIF	AIF	POF	COF	SIZE	REWORK
20	60	50	50	1	10	20000	148.20
20	60	50	50	2	10	20000	145.74
20	60	50	50	3	10	20000	143.09
20	60	50	50	4	10	20000	139.94
20	60	50	50	5	10	20000	136.00
20	60	50	50	6	10	20000	130.99
20	60	50	50	7	10	20000	124.69
20	60	50	50	8	10	20000	116.89
20	60	50	50	9	10	20000	107.38
20	60	50	50	10	10	20000	95.98

RESPONSE OF REWORK WITH INCREASE IN COF

MHF	AHF	MIF	AIF	POF	COF	SIZE	REWORK
20	60	50	50	10	5	20000	35.39
20	60	50	50	10	10	20000	95.98
20	60	50	50	10	15	20000	140.12
20	60	50	50	10	20	20000	161.74
20	60	50	50	10	25	20000	166.78
20	60	50	50	10	30	20000	165.00
20	60	50	50	10	35	20000	164.28
20	60	50	50	10	40	20000	168.10
20	60	50	50	10	45	20000	177.09
20	60	50	50	10	50	20000	191.22

REWORK should increase with increase in size which is shown below. Hence, the model is valid.

RESPONSE OF REWORK WITH INCREASE IN SIZE

MHF	AHF	MIF	AIF	POF	COF	SIZE	REWORK
20	60	50	40	10	10	20000	126.97
20	60	50	40	10	10	21000	133.10
20	60	50	40	10	10	22000	139.08
20	60	50	40	10	10	23000	144.93
20	60	50	40	10	10	24000	150.63
20	60	50	40	10	10	25000	156.19
20	60	50	40	10	10	26000	161.61
20	60	50	40	10	10	27000	166.88
20	60	50	40	10	10	28000	172.01
20	60	50	40	10	10	29000	177.00

CONCLUSION AND FUTURE WORK

The proposed model can be used by developers to predict maintainability efforts at design phase itself and thus can make required changes before the software is actually coded. The Mean Absolute Error for this Artificial Neural Network is 0.1799. Thus it is appropriate to predict the maintainability efforts required. The analysis results show which factor is more important and by what weight. Thus developer can choose certain object oriented mechanisms over other. MOOD metrics gives complete project level results. Thus, this model can be used by project managers for analyzing various design models.

The data-set can be extended by developing tool to measure MOOD values and further more metrics relevent to maintainability can be incorporated.

REFERENCE

[1] Fernando Brito e. Abreu, M. Goulão and R. Esteves, "Toward the design quality evaluation of object-oriented software systems." Proceedings of the 5th International Conference on Software Quality, Austin, Texas, 1995.

[2] Fernando Brito e. Abreu, "Design metrics for object oriented systems", 1996.

[3] Fernando Brito e. Abreu and Walcélio. Melo, "Evaluating the impact of Object-Oriented Design on software quality", Originally published in Proceedings of the 3rd International Software Metrics Symposium (METRICS'96), IEEE, Berlin, Germany.

[4] Fernando Brito e. Abreu, R. Esteves and M. Goulão, "The design of Eiffel programs: quantitative evaluation using the MOOD metrics", originally published in proceedings of tools'96 usa, santa barbara, california, 1996.

[5] A. Hincheeranan and W. Rivepiboon, "A maintainability estimation model and tool" , pp.-143-146, International Journal of Computer and Communication Engineering, Vol. 1, No. 2, 2012.

[6] K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, "Application of artificial neural network for predicting maintainability using object oriented metrics" , World Academy of Science, Engineering and Technology Vol:2, 2008.

[7] Y. Dash, S. K. Dubey and A. Rana, "Maintainability prediction of object oriented software system by using artificial neural network approach", IJSCE ISSN: 2231-2307, Volume-2, Issue-2, 2012.

[8] A. Kumar, A. Kalia and H. Singh, "Metrics identification for measuring object oriented software quality", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-5, 2012.

[9] G. Kaur and M. Aggarwal,"A fuzzy model for evaluating the maintainability of object oriented system using MOOD metrics", IJCSCE, Volume 3, issue1, 2014.

[10] K. Kaur and A. Sami, "A maintainability estimation model and metrics for object-oriented design (MOOD)", ISSN: 2278-1323 International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, No 5, 2013.

[11] S.W.A. Rizvi and R.A. Khan, "Maintainability estimation model for object- oriented software in design phase (MEMOOD)", journal of computing, volume 2, issue 4, 2010.

[12] H. Singh and A. Kumar, "A novel approach to enhance the maintainability of object oriented software engineering during component based software engineering", International Journal of Computer Science and Mobile Computing, Vol.3 Issue.3, pg. 778-786, 2014.

[13] M. Saini and M. Chauhan, "A roadmap of software system maintainability models", International Journal of Software and Web Sciences 3(2), pp. 69-73, 2012.

[14] M. Sarker, "An overview of Object Oriented Design Metrics", Department of Computer Science, Umeå University, Sweden, 2005.

[15] J.T.S. Quah and 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.

[16] R. D. Bankar, S. M. Datar and D. Zweig, " Software complexity and maintainability".

[17] R. Harrison, S.J. Counsell and R. V. Nithi, "An evaluation of the MOOD set of object oriented software metrics",University of Southampton, 1998.