A Novel PCA based Multi-layer perceptron algorithm for Maintainability Prediction

Deeksha Datyal[1], Aman Kaushik[2], Abhishek Tomar[3]

[1]Research Scholar, [2]Assistant Professor, [3]Assistant Professor
Department of Computer Science, Baddi University of Emerging Sciences and Technology
Baddi, Solan (H.P), India

Abstract: *Software Engineering has attracted the interest of the researchers all around the world in the recent years. Various software metrics becomes the index for measuring the quality of any software engineering. Software maintainability is one of the important metrics which needs to predict in advanced for better performance during the SDLC cycle. Various algorithms have been attempted in the past for the same and their performance has been measured. A classification algorithm such as KNN has been the one of the primary algorithms. This paper implements a novel PCA based Multi-layer perceptron algorithm for maintainability prediction. The maintainability is predicted and compared to that of the actual values and accuracy, precision and recall values are calculated. It is found that our algorithm performs quite well and gives encouraging results*
.

Keywords:*Software Maintainability, Principal Component Analysis (PCA), Multi-Layer Perceptron (MLP), Software Development Life Cycle Model (SDLC).*

I. INTRODUCTION

Software maintainability is the ease with which a software system can be modified, is a significant software quality attribute. Intrinsically associated with this quality attribute is the maintenance process that is being acknowledged to signify the majority of the costs of a Software Development Life-Cycle (SDLC). Therefore, the maintainability of a software system can considerably impact software costs. The changes in the software is necessary to convene the changing requirements of customers which may arise due to many reasons like change in the technology, introduction of new hardware or enhancement of the features provided etc. Producing software which does not need to be changed is not only impractical but also very uneconomical. This method of varying the maintenance of software is known as maintenance. The amount of resource, effort and time spent on software maintenance is much more than what is being spent on its development. Thus, developing the software which is simple to sustain and can save big costs and efforts. The main approaches in controlling maintenance cost are to monitor software metrics during the development phase.

Prediction, basically called as estimation, is a significant part of project planning. Estimates can be made for projects or

processes as well as products. When these are made for projects, these are called effort estimates and the process is called effort estimation or software cost estimation. When prediction is ready to a maintenance process, the means of getting such predictions is called maintenance cost prediction or maintenance project effort estimation. In addition, estimates of quality attributes provides a measurable value of the quality of the attribute that a software product possesses. The main centre of this paper is on the estimation of the quality attribute of maintainability. Software maintenance is defined as "the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment". From the definitions it is clear that maintenance is the process performed as part of the SDLC whereas maintainability is the quality attribute associated with the software product. A software maintainability prediction model can give a means to better manage their maintenance resources and also to adopting a defensive design. This can then help in reducing the maintenance effort and therefore, reducing the overall cost and time spent on a software project.

A. Uses of accurate prediction of software maintainability

- It helps project managers in comparing the productivity and costs among different projects.
- It provides managers with information for more effectively planning the use of valuable resources.
- It helps managers in taking important decision regarding staff allocation.
- It guides about maintenance process efficiency.
- It helps in keeping future maintenance effort under control.

II. RELATED *WORK*

YashTashtousget al. [1] analyzed the 5 public domain software defect datasets provided by NASA

IV & V Facility and Metrics Program (MSP) repository to find correlation between different software complexity metrics. It was found that Cyclomatic Complexity has strong correlation with Halstead Complexity and LOC. Halstead Complexity was found to have strong correlation with Cyclomatic Complexity but weak correlation with LOC. Cyclomatic Complexity and Halstaed Complexity are strongly correlated and used together, Cyclomatic Complexity for measuring control flow and Halstead Complexity for measuring data flow.

Hairuddin and Elizabeth [2] proposed a maintainability model in this paper. Moreover several parameters such as Readability, Standardization, Programming Language,Modularity,Level of Validation and Testing has been studied in order to calculate the maintainability of software systems,Complexity and Traceability can be taken in account.

Fioravanti and Nesi[3] presented a model for effort estimation/prediction of adaptive maintenance. It assumed that the system efforts are generally spent on doing various operations such as insertion, addition, subtraction, deletion of fact, changes of system code etc. This metric is useful for various software projects and validated for predicting adaptive maintenance. The results show that the proposed model is performsbetter as compared with other traditional model.

Bandiniet al.[4] this paper measured the 3 independent factors, namely; design complexity, maintenance task and programmer's ability to predict the maintenance performance for object-oriented systems. To measure design complexity, Perfective and corrective maintenance has been chosen to signify maintenance task. Correlation analysis was carried out to summarize that the selected attributes is capable to estimate the maintenance efforts of the systems.

Ahnet al. [5] presented a model of measuring software maintenance project effort estimation and this proposed approach is depending on the function point measure and 10 maintenance productivity factors. These factors are categorized as engineer's skills, technology characteristics, and maintenance environment.

The estimation was prepared by **Ardimento** et al. [6] that if a component is difficult to understand then it will be difficult to maintain it and advised the trial usage of component before adopting it for the application.

Riazet al. [7] in this paper, the effect of systematic survey on maintainability predictions and metrics has been proposed. Out of 710 reviews 15 were selected for systematic review. Typically, model has been designed for the prediction of maintainability was depends on algorithmic techniques but that algorithm is to be applied to which maintenance type, this type of distinction is not available. The researchers proposed several model for designging which are more efficient, robust and reliable.

Thwin and Quah[8] used neural networks to build software quality prediction models. They proposed that maintainability can be estimated with the help of fuzzy model. They also proved empirically that the integrated measure of maintenance obtained from this fuzzy model has strong correlation with maintenance.

Zhou and Leung[9]multivariate adaptive regression splines (MARS) has been utilized for estimating the maintainability of object-oriented software. They contrast the prediction accuracy of proposed model with 4other prevailing models: MLR, SVR, ANN, and regression tree (RT) and stated that MARS is best model to be used as far as maintainability of prediction is concerned.

Hu and **Zhong[10]**proposed a model depending on neural network to predict software module risk. Software quality has been estimated by utilizing the learning vector quantization network.

Arvindar et al. [11] predicted the software maintenance effort by application of diverse soft computing approach. Two software products were taken as dataset and they observed that proposed approach is useful for the creation of accurate models to consider the maintenance effort. In their analysis maintenance effort is selected as dependent variable and eight OO metrics as independent variable.

III. *PROBLEM* FORMULATION

Maintainability prediction in software industry is a continuous process and affects almost all segments of the Software Development Life Cycle (SDLC) and even after that in terms of maintainability and Quality of service. Maintainability prediction can include decisions regarding the issue of funds, deployment of manpower for development team, testing manpower regularization and many others. While economic decisions are taken on the basis of software cost estimation models, on the other hands, decision regarding the regularization of software testing team depends a lot on the number of reported bugs and fault prediction. Reliability of software is a direct consequence of the probability of fault occurrence. Thus fault prediction and maintainability of the software are highly correlated term. All these are known as external metrics which needs to be predicted in advance for optimal maintainability prediction process.

Over the recent years, it has been found that all these factors depends a lot on some of the factors or parameters of the software, more commonly known as software internal metrics. Various researches has been done on finding the influence of these internal metrics on the external metrics. A strong correlation is found to exist with some of the internal software metrics. Apart from that there seems to exist some strong correlation among the internal software metrics itself. This thesis aims at finding these correlations using a novel Principle Component Analysis based feature selection methodology so that they can be utilized for improved maintainability prediction using Multi-Level Perceptron.

IV. *PROPOSED* METHODOLOGY

The process involved in the thesis can be briefly summarised as:

Step 1: Finding the various factors affecting maintainability prediction in SDLC.

Step 2: Finding the correlation between internal and external software metrics.

Step 3: Finding cross correlation among the metrics.

Step 4: Development of Principle Component Analysis for Feature Selection.

Step 5: Using Feature selection for finding significant features from vast number of attributes.

Step 6: Development of Hybrid PCA-MLP algorithm for improved maintainability prediction.

Step 7: Maintainability prediction parameter prediction.

Step 8: Dataset will be used from Promise Repository of NASA.

Step 9: KLOC (Kilo Lines of Code), McCabe's Cyclomatic Complexity, Halstead Volume, Program Length, Number of Comments, Effort Estimate, etc.

A. Performance Parameters

These are the performance parameters on which our algorithm accuracy, efficiency and complexity would be measured.

i) Accuracy

- *Accuracy*: is the proportion of the total number of predictions that were correct
- Error rate (misclassification rate)= 1 – AC

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

ii) Precision

- Precision or Confidence (as it is called in Data Mining) denotes the proportion of Predicted Positive cases that are correctly Real Positives.
- Precision is defined below:

$$\textbf{Precision} = \text{Confidence} = \text{tpa} = \frac{tp}{pp} = \frac{TP}{PP} = \frac{TP}{TP + FP}$$

iii) Recall

- Recall or Sensitivity (as it is called in Psychology) is the proportion of Real Positive cases that are correctly Predicted Positive.

$$\textbf{Recall} = \text{Sensitivity} = \text{tpr} = \frac{tp}{rp} = \frac{TP}{RP} = \frac{TP}{TP + FN}$$

iv) F-Score/Measure

- The F-Measure computes some average of the information retrieval precision and recall metrics.
- The F-measure of cluster j and class i is defined as follows:

$$F_{ij} = \frac{2 * \text{Recall}(i,j) * \text{Precision}(i,j)}{\text{Recall}(i,j) + \text{Precision}(i,j)}$$

- F-measure is computed using the harmonic mean:
  - Given n points, x1, x2, …, xn, the harmonic mean is:
    - $\frac{1}{H} = \frac{1}{n}\sum_{i=1}^{n}\frac{i}{x_i}$
- So, the harmonic mean of Precision and Recall

$$\frac{1}{F} = \frac{1}{2}\left(\frac{1}{R} + \frac{1}{p}\right) = \frac{P + R}{2PR}$$

B. Comparison Parameter's

These parameters help us in comparing our algorithm with other algorithms and techniques which are used for software quality prediction.

i) Confusion Matrix

- A **table of confusion** (sometimes also called a **confusion matrix**).
- Is a table with two rows and two columns that reports the number of *false positives*, *false negatives*, *true positives*, and *true negatives*.

Table 1 –Confusion Matrix

		Predicted Condition	
	Total Population	Predicted Condition (Positive)	Predicted Condition (Negative)
Actual Condition	Actual Condition (Positive)	A: True Positive	B: False Negative
	Actual Condition (Negative)	C: False Positive	D: True Negative

C. Multilayer perceptron (MLP)

A multilayer perceptron (MLP) is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate outputs. A multilayer perceptron composed of many layers of nodes in a directed graph, with each layer associated to the next one. Excluding for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called back propagation for training the network. Multilayer perceptron is a modification of the standard linear perceptron and can differentiate data which are not linearly separable. The multilayer perceptron categorised of three or more layers of nonlinearly-activating nodes and is hence considered a deep neural network. Every node in one layer connects with a certain weight to each node in the following layer.

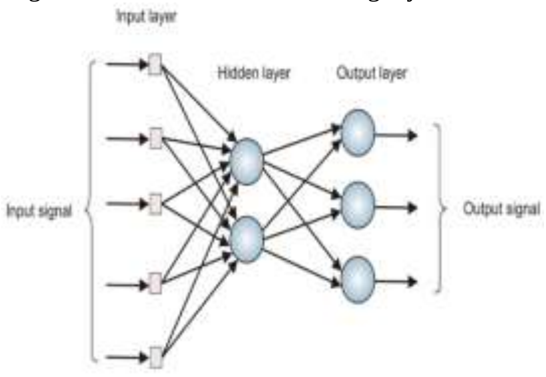

Fig 1: Multilayer perceptron Model

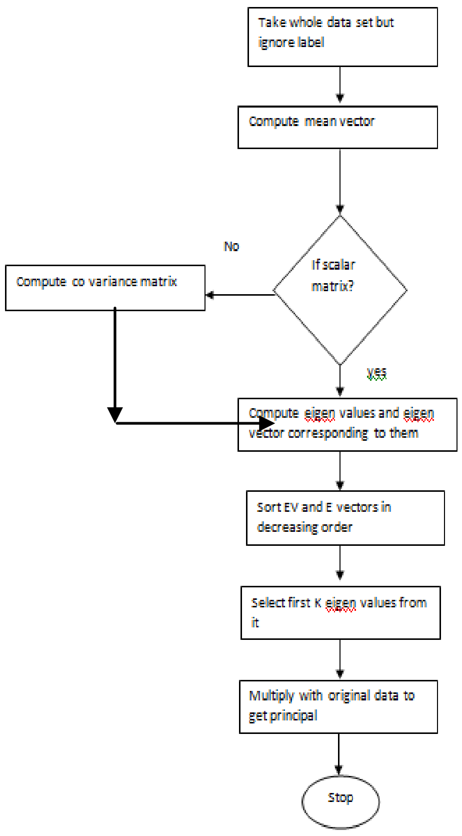
Take whole data set but ignore label
Compute mean vector
No
If scalar matrix?
Compute co variance matrix
yes
Compute eigen values and eigen vector corresponding to them
Sort EV and E vectors in decreasing order
Select first K eigen values from it
Multiply with original data to get principal
Stop

Fig 2: Multi layer perception Flow Chart

D. Principal Components Analysis

In principal components analysis (**PCA**) and factor analysis (**FA**) one desires to extract from a set of *p* variables a reduced set of *m* components or factors that accounts for most of the variance in the *p* variables. On the other hand, we desire to minimize a set of *p* variables to a set of *m* underlying super ordinate dimensions. Every factor is calculated as a weighted sum of the *p* variables. The i^{th} factor is thus

$$F_i = W_{i1}X_1 + W_{i2}X_2 + ... + W_{ip}X_p$$

One can state every of the *p* variables as a linear combination of the *m* factors,

$$X_j = A_{1j}F_1 + A_{2j}F_2 + ... + A_{mj}F_m + U_j$$

Where U_j is the variance that is unique to variable *j*. This process is attained by estimating a matrix of coefficients whose columns are called eigen vectors of the variance-covariance or of the correlation matrix of the data set. PCA is a method to study the structure of the data, with emphasis on determining the patterns of co-variances among variables. Thus, PCA is the study of the structure of the variance-covariance matrix. In practical terms, PCA is a method to identify variable or sets of variables that are highly correlated with each other. Some basic consequences of the process are that:

- All original variables are involved in the calculation of PC scores
- The sum of variances of the PC's equals the sum of the variances of the original variables when principle component analysis is depending on the variance-covariance matrix, or the sum of the variances of the standardized variables when principle component analysis is depending on the correlation matrix.
- There are p eigen values (p=number of variables in the data), everyone is connected with one eigenvector and a PC. Therefore, the sum of eigenvalues depending on the variance-covariance matrix is equal to the sum of variances of the original variables.

Principle component analysis is depending based on the correlation matrix is equal to using principle component analysisdepends on the variance-covariance of the standardized variables. Due to the standardized variables have variance equals to 1, the sum of eigenvalues is p, the number of variables.

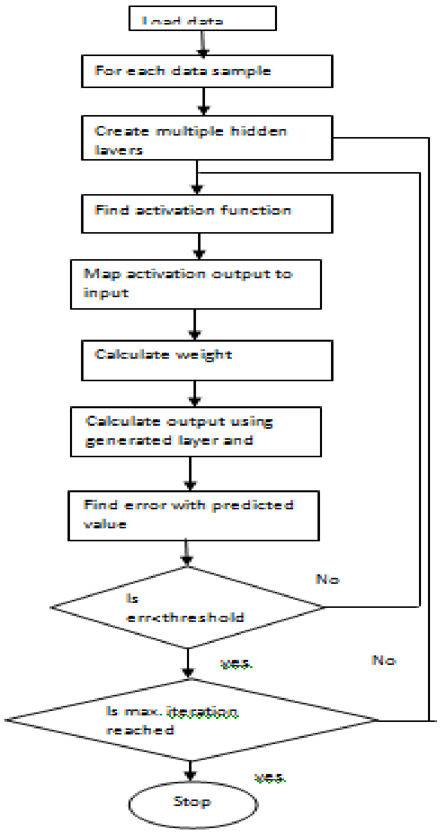
Load data
For each data sample
Create multiple hidden layers
Find activation function
Map activation output to input
Calculate weight
Calculate output using generated layer and
Find error with predicted value
Is err<threshold
No
yes
No
Is max. iteration reached
yes
Stop

Fig.2: PCA Flow Chart

V. RESULTS

First Correlation analysis is done on the software maintainability data and the correlations are found out between various metrics. The results of the correlation are shown below.

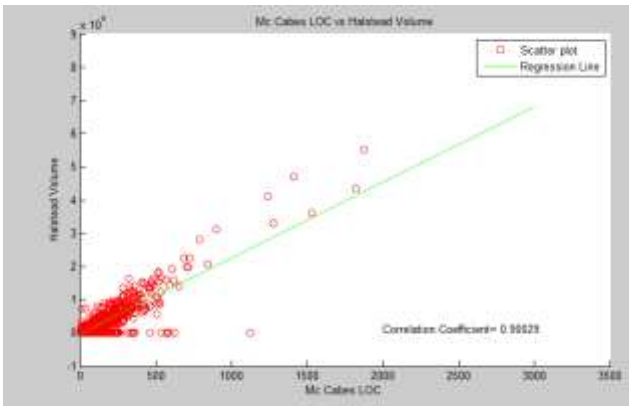

Fig. 3(a): Correlation between Mc Cabes LOC vs Halstead Volume

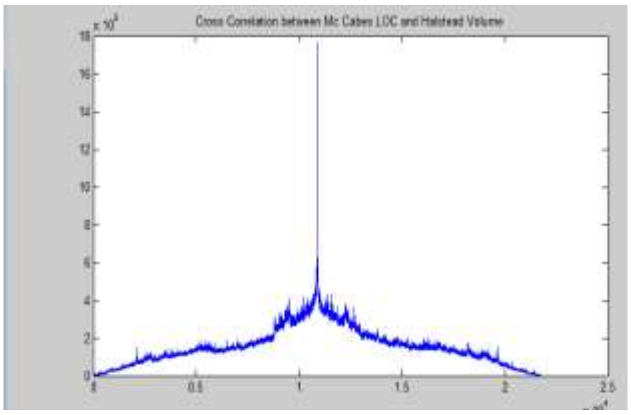

Fig. 3(b) :Cross-Correlation between Mc Cabes LOC and Halstead Volume

The PCA is applied using these correlation values and the results are shown below

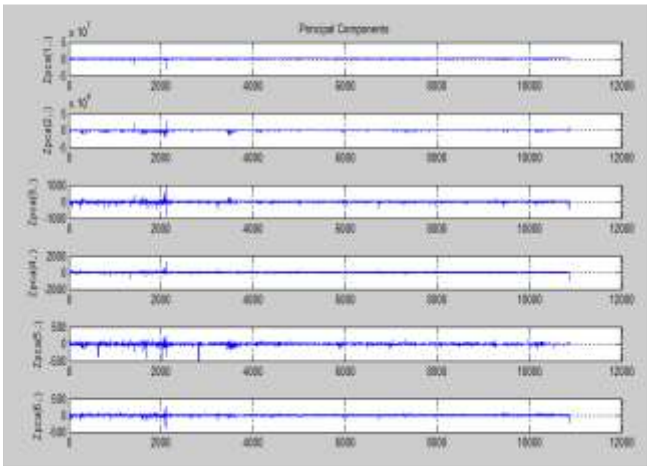

Fig. 4(a): variance of extracted principle components.

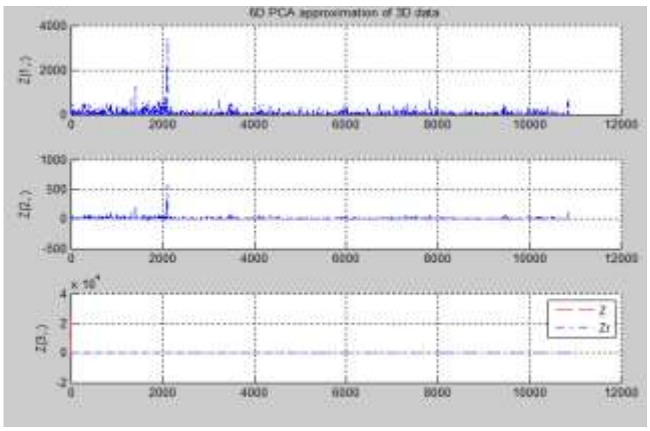

Fig.4(b):6 extracted PCA components and correlations with time.

After application of PCA and MLP, the following confusion matrix is obtained:

Confusion matrix=$\begin{bmatrix} TP & FP \\ TN & FN \end{bmatrix}$

Where value for TP= 2106, FP=411, TN=8340, FN=28

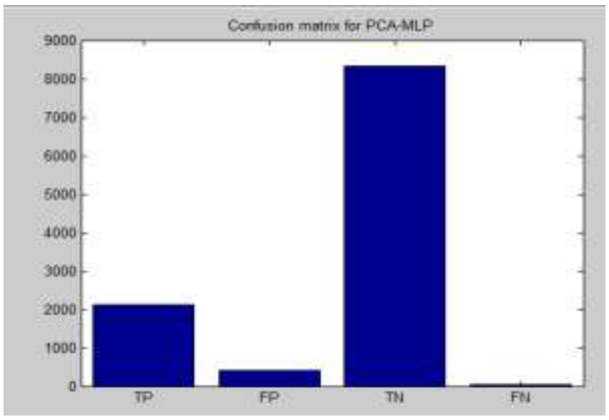

Fig.5:Confusion Matrix

In order to calculate the values for accuracy, prediction, recall, f-score values generated in Confusion Matrix are applied to their respective formulas i.e.

Accuracy=$\frac{TP+TN}{TP+FP+TN+FN}$ ----- (1)

Precision=$\frac{TP}{TP+FP}$ ----- (2)

Recall=$\frac{TP}{TP+FN}$ ----- (3)

Fscore=$\frac{2*Precision*Recall}{Precision+Recall}$ ----- (4)

The results have also been compared to that of an Artificial Neural Network and a comparison of our proposed algorithm with ANN in terms of accuracy, precision, recall and F-score is presented below.

Table 2:Comparing Performance of ANN and Proposed Algorithm

	Accuracy	Precision	Recall	F-Score
Proposed Algorithm	95.9669	83.671	98.6781	90.5612
ANN	94.9564	80.3388	98.4296	88.4688

As observed the performance of our proposed algorithm on the same data set is better in terms of accuracy, precision and F-Score while the recall remains almost same as compared to ANN. This can be attributed to the fact that our algorithm is more or less same in terms of rejection rate and feature selection doesn't have much role in rejection rate thus the Recall value is almost unaffected.

VI. CONCLUSION AND FUTURE SCOPE

Predicting Software metrices in advance is an important trend these days of which helps in boosting the performance of software development process. There are two types of metrices namely; external and internal. The dependencies among the external and internal metrices needs to be understood completely in order to enhance system performance. This paper proposed an approach of predicting software maintainability from various metrices. The correlation between various metrices has been found out and the regression line is plotted. Also the pearson correlation coefficient is calculated and shown. A hybrid Principle Component Analysis-MultiLayer Perceptron model was proposed and utilized for the prediction of software maintainability. The PCA is first applied to reduce the dimensions of the data and extract meaningful features and the improved dataset is then utilized by MLP for maintainability prediction. The performance is found in terms or accuracy, precision, recall and F-Score. The results are found to be quite encouraging.
In future other algorithms like Linear Discriminant Analysis can be applied for dimensionality reduction. Also Neural Networks can be utilized for prediction. The developed algorithm can be tested on other datasets and hybrid algorithms can be developed for the same.

REFERENCES

[1] Tashtoush, Yahya, Mohammed Al-Maolegi, and BassamArkok. "The correlation among software complexity metrics with case study." *arXiv preprint arXiv:1408.4523* (2014).

[2] Khairuddin, H., Elizabeth, K., 1996. A Software Maintainability Attributes Model, Malaysian Journal of Computer Science, Vol. 9, Issue 2, pp: 92-97

[3] Fioravanti, F., Nesi, P., 2001. Estimation and Prediction Metrics for Adaptive Maintenance Effort of Object - Oriented Systems, IEEE Transactions on Software Engineering, Vol. 27, Issue 12, pp: 1062–1084.

[4] Bandini, S., Paoli, F. D., Manzoni, S., Mereghetti, P., 2002. A support system to COTS based software development for business services , Proceedings of the 14th International Conference on Software Engineering and Know ledge Engineering, Ischia, Italy, Vol. 27, pp: 307–314.

[5] Ahn, Y., Suh, J., Kim, S., Kim, H., 2003. The Software Maintenance Project Effort Estimation Model Based on Function Points, Journal of Software Maintenance: Research and Practice, Vol. 15, Issue 2, pp: 71-85.

[6] Ardimento, P., Bianchi, A., Visaggio, G., 2004. Maintenance-oriented Selection of Software Components, Proceedings of 8th European Conference on Software Maintenance and Reengineering, pp: 115 –124.

[7] .Riaz, M., Mendes, E., Tempero, E. D.: A Systematic Review of Software Maintainability Prediction and Metrics. In: ESEM 2009, 2009, pp. 367-377.

[8] 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.

[9] Zhou Y, Leung H (2007) Predicting object-oriented software maintainability using multivariate adaptive regression splines. J SystSoftw 80(8):1349–1361. doi:10.1016/j.jss.2006.10.049.

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

[11] 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.

[12] Kajko-Mattsson, M., Canfora, G., Chorean, D., van Deursen, A., Ihme, T., Lehmna, M., Reiger, R., Engel, T., Wernke, J., 2006. A Model of Maintainability – Suggestion for Future Research, Proceedings of International Multi-Conference in Computer Science & Computer Engineering (SERP’06), pp: 436-441.

[13] Grover, P. S., Kumar, R., Sharma, A., 2007. Few Useful Considerations for Maintaining Software Components and Component -Based Systems. ACM SIGSOFT Software Engineering Notes, Vol. 32, Issue 4, pp: 1-5.

[14] Kumar, Avadhesh, Rajesh Kumar, and P. S. Grover. "An evaluation of maintainability of aspect-oriented systems: a practical approach." *International Journal of Computer Science and Security* 1.2 (2007): 1-9.

[15] Zavvar, Mohammad, and FarhadRamezani. "Measuring of Software Maintainability Using Adaptive Fuzzy Neural Network." *International Journal of Modern Education & Computer Science* 7, no. 10 (2015).

[16] Reddy, B. Ramachandra, SahilKhurana, and AparajitaOjha. "Software Maintainability Estimation Made Easy: A Comprehensive Tool COIN." In *Proceedings of the Sixth International Conference on Computer and Communication Technology 2015*, pp. 68-72. ACM, 2015.