An Empirical Study to Redefine the Relationship between Software Design Metrics and Maintainability in High Data Intensive Applications

Ruchika Malhotra and Anuradha Chug

***Abstract*—Software maintainability is defined as the ease with which modifications could be made in to the software once it is delivered to the customer. While evaluating the quality of the software product, software maintainability is one of the most important aspects and it is desirable that the software should be designed and coded in such a way that it becomes more maintainable. Tracking the maintenance behavior of the software product is very complex and widely acknowledged by the researchers. We can accurately measure 'maintainability' of any software once it comes into operations but it would be too late by then, hence much has been examined in literature to measure the maintainability before software start operations by making use of software design metrics. It has proved empirically many times that there exists strong relationship between software design metrics and its corresponding maintainability. However, the framework and reference architecture in which the softwares are developing now days have changed dramatically as they make heavy use of databases. There is a strong need to re-define the relationship between software design metrics with subsequent maintainability in this changed scenario. In an attempt to address this issue quantitatively, we have proposed new suite of metrics by the induction of two new metrics which are more important and meaningful in data intensive applications. To analyze the proposed metric suite, their values are computed on five real-life applications which make use of databases with a great deal. The result shows that proposed new metrics suite is very effective indicator of software maintainability in the environment which provide remote connections to the server for accessing large database files. Based on the results it can be reasonably claimed that new metrics suite proposed in the current study would be able to predict software maintainability more precisely and accurately for those applications which makes heavy use of databases during operations.**

***Index Terms* - Object Oriented Metric, Maintainability predictions, Software quality, Empirical validation, Database design metrics**

I. Introduction

The modifications in the software are required to meet the changing requirements of customers which may arise due to many reasons such as change in the technology, introduction of new hardware or enhancement of the provided features etc. Producing software which is not required to be changed is not only impractical but also very uneconomical. This process of changing the software which has been delivered previously is called software maintenance and the ease with which it could be achieved is defined as software maintainability [1].

Manuscript received Aug 07, 2013; Revised Aug 17, 2013.

Ruchika Malhotra is with Dept of Software Engineering, Delhi Technological University, Delhi, India (ruchikamalhotra2004@yahoo.com)

Anuradha Chug is with USICT, GGSIP University, Dwarka, New Delhi, India (corresponding author +91-11-25302715/25302318; e-mail: a_chug@yahoo.co.in).

It has been observed that the amount of resource, effort and time spent on software maintenance is much more than what is being spent on software development [2]. Thus, producing software that is easy to maintain may potentially save large costs. One of the most common approaches for controlling maintenance cost is by utilizing software design metrics during the development phase [3]. Various metrics have been proposed in the literature along with their corresponding impact on software maintainability. In this paper we have selected significant subset of seven metrics proposed in literature and added two new metrics as they found to be more impactful on software maintainability in current scenario where applications are intensely using databases during operations. The main purpose of this paper is twofold, firstly:

(a) Review the role of various metrics which are proposed in literature on software maintainability and secondly

(b) To purpose and empirically validate a new suite of metrics with the induction of two new metrics which have larger impact on software maintainability in current scenario where more data intensive application development is in progress.

For empirical validation of the proposed metrics suite we have collected the data from five windows based and web based applications. All applications were based on object oriented (OO) methodologies and developed in Microsoft Visual Studio using C# language and exploit the use of databases. Values of proposed metrics are collected for every class of each application. The independent variables are nine design metrics used to measure OO feature and database design feature present in the code. Dependent variable in our study is ‘Change’ and counted as number of lines of source code added, deleted or modified during operations. Use of Artificial Neural Network (ANN model) for the prediction of maintainability has reported as best fitting machine learning model [8, 9, and 10]; therefore it is used in current study for building the prediction model using data points collected from all five real-life application for new proposed metrics suite. We hope that while designing and coding, developers can analyze and predict maintainability more precisely with the help of proposed metrics suite. Detecting the ‘Maintainability’ in early phases of Software Development Life Cycle (SDLC) would ensure that correction has no side effects on dependent modules. Further, developers can judge if the application is maintainable or not. This in itself would save time and money for the organization responsible for developing and deploying.

The remainder of the paper is organized as follows: Section II summarizes related work. Section III elucidates empirical data collection method. In Section IV, independent and dependent variables selected in the study are discussed. Section V enumerates research methodology. In Section VI explores characteristics of the collected data using descriptive statistics. The results and analysis is presented in Section VII. Section VIII states threat to validity and finally section IX concludes the paper with future directions.

II. RELATED WORK

There are several models and metrics proposed in literature to predict the maintainability of the software. Chidamber et al. [4] outlined some initial proposals for language-independent OO design metrics in 1991 and proposed six object oriented design metrics to predict the maintainability of OO systems. This suite is further expanded in 1994 and the metric suite was tested on systems developed in C++ and Smalltalk(TM) by Chidamber et al [6]. Two more metrics were added in to existing C&K metrics suite by Li et al [5 and 7]. The proposed metrics suite became quite popular and evaluated analytically in several studies by many researchers including Dagpinar et al. [8], Thwin et al [9], Aggarwal et al [10], Koten et al [11], Zhou et al [12], Elish et al [13] and Kaur et al [14] which were conducted to validate C&K metrics suite. Many prediction models were built using statistical algorithms as well as machine learning algorithms. Neural Network was used by Thwin et al [9], Bayesian Belief Network (BBN) was used by Koten et al [11], Multivariate Adaptive Regression Splines (MARS) was used by Zhou et al [12] and TreeNets was used by Olish et al [13] to build models for the prediction of software maintainability using OO design metrics suite. Ping [15] suggested that more quantitative approach to measure maintainability should be adopted. Jin et al [16] have successfully applied use of Support Vector Machine (SVM) to predict software maintainability using OO metric suite. Recently Malhotra et al [17] have proposed the use of Group Method of Data Handling (GMDH) to predict software maintainability more concisely and precisely. However, there is strong need to redefine the metrics suite used for prediction of software maintainability which is more relevant in data intensive applications. In an attempt to address this problem, the current study was undertaken to ascertain the cumulative effects by the attributes of object oriented paradigm and database design characteristics on software maintainability. To achieve this aim, authors have proposed and validated new metrics suite which is found to be more appropriate and precise in predicting maintainability for data intensive applications.

III. EMPIRICAL DATA COLLECTION

Five customized softwares were selected for empirical validation out of which three were window based applications and two were web based applications. All five applications were data intensive and sophisticated as they concentrate on collecting, maintaining, indexing and computing data from database. Their functioning is described in brief as follows:

- FLM is a customize software to handle the File Letter Monitoring System. (FLM System)
- EASY is a web portal for an Educational Institute which provide study material online. (EASY System)
- SMS system is Student Management System which maintains the record of students & teacher for private educational institute. (SMS System)
- IM System is Inventory Management System which maintains inventory of company at different branch offices in different cities. (IMS System)
- ABP system is Angel Bill Printing software maintains fully editable items list by client itself with generation of a common bill format. (ABP System)

To measure the maintainability we first found out the “change effort” which is defined as amount of average efforts required to add, change or delete lines of source code. To measure the various features of object oriented paradigm such as data hiding, inheritance, cohesion, coupling, memory allocation etc different metrics were carefully selected. We reviewed various metrics proposed in literature and compiled in Malhotra et al [17] and Aggarwal et al [18]. We selected only those software design metrics which were proven to be strongly associated with software maintainability. Further two more metrics were proposed (described in section 4) in the study as they found to be more meaningful in data intensive applications. Two versions of each of the software were taken and analyzed for “Changes” made into it. Changes were counted in terms of added, deleted and modified lines in the recent version with respect to the previous version. Addition and deletion of line in source code was counted as one change whereas modification was counted as two (one line added, one line deleted) changes. Selected Metrics as described in section 4 were collected for these five web and window based applications. Tool was created in visual studio which collected the values of selected metrics which were further used for constructing prediction model to drive object oriented software maintainability.

IV. INDEPENDENT AND DEPENDENT VARIABLES

The selected five applications namely FLM, EASY, SMS, IMS and ABP were developed in Visual Studio 2010. To measure various attributes of OO paradigm such as size, coupling, cohesion and inheritance, one metric alone would not be sufficient hence total nine metrics were selected and compiled in Table I and Table II. Five metrics namely Maintainability Index (MI), Cyclomatic Complexity (CC), Depth of Inheritance (DOI), Coupling between Objects (CBO) and Lines of Code (LOC) were retrieved from the Visual Studio by considering all applications one by one in the .net environment. For description of these metrics please refer Table I wherein the metrics mentioned were calculated from the intermediate language code generated while compiling the project. In the next stage of our research work, we have created a tool in visual studio to collect four metrics namely Code to Comments Ratio (CCR), Number of Data Base Connections (NODBC), Weighted Methods per Class (WMC) and Lack of Cohesion of Methods (LCOM). First two metrics namely CCR and NODBC have been proposed for the first time in the current study and as the part of our research project, we proposed that these two metrics carries more impact on software maintainability in database intensive applications. Apart from database design, all aspects of object oriented design such as coupling, cohesion and inheritance are equally important. Since complexity and cohesion were missing in Table I, two metrics WMC and LCOM were taken from C&K Metrics suite [4-7] to cover complexity and cohesion respectively. For description of these metrics please refer Table II.

Next, source codes of two versions (original version and modified version) were taken for each application. Values of first five metrics as described in Table I were obtained through .net and values of remaining four metrics as described in Table II were obtained through the Tool we created in first phase. We investigated source code of both versions (original version and modified version) for each

TABLE I : METRICS COLLECTED FOR OBJECT ORIENTED SOFTWARE

S no	Metric Name	Description
1	Maintainability Index (MI)	Calculates an index value between 0 and 100 that represents the relative ease of maintaining the code.
2	Cyclomatic Complexity (CC)	Measures the structural complexity of the code. It is created by calculating the number of different code paths in the flow of the program module. A program that has complex control flow will require more tests to achieve good code coverage and will be less maintainable.
3	Depth of Inheritance (DIT)	Indicates the number of class definitions that extend to the root of the class hierarchy. The deeper the hierarchy the more difficult it might be to understand where particular methods and fields are defined or/and redefined.
4	Coupling Between Object classes	Count the number of other classes to which it is coupled. To measures the coupling to unique classes through parameters, local variables, return types, method calls, generic or template instantiations, base classes, interface implementations, fields defined on external types, and attribute decoration.
5	Lines of Code (LOC)	Indicates the approximate number of lines in the code. The count is based on the Intermediate Language (IL) code and is therefore not the exact number of lines in the source code file. A very high count might indicate that a type or method is trying to do too much work and should be split up. It might also indicate that the type or method might be hard to maintain.

application to calculate the amount of 'change' made in each class. It is calculated manually by comparing the recent version of source code for class with the previous version of the same class. Difference is observed in terms of lines of source code added, deleted or modified for each class. The values of collected metrics were finally compiled class wise with respective values of 'change' made in that class in the recent version and combined respectively to generate data points. We have compiled all the classes for each application to generate data points. For FLM System, EASY System, SMS System, IMS System and for ABP System the number of data points are 233, 292, 129, 96 and 114 respectively. Same methodology is adopted in Zhou et al [19].

TABLE II : METRICS COLLECTED BY TOOL FOR SELECTED WEB AND WINDOW BASED APPLICATIONS

SNo	Metric Name	Description
1	CCR (Comments to Code Ratio)	Ratio of number of comments lines to number of code lines in C# code file. Number of comment lines to the tatal number of lines in the source code were compared to find the values of this metric.
2	NODBC (Number of Data Base Connections)	Number of Data Base Connection is a measure to count number of times database connection were made. It is counted by counting 'Open()' function call for database connection in the source code.
3	WMC (Weighted Methods per Class)	The sum of Mc Cabes's cyclomatic complexities of all local methods in a class. Let a class K_1 with method M_1… M_n that are defined in the class. Let C_1…….Cn be the complexity of the methods.
4	LCOM (Lack of Cohesion of Methods)	The number of disjoint sets of local methods. Each method in a disjoint set shares at least one instance variable with at least one member of the same set.

V. RESEARCH METHODOLOGY

Recent research activities have revealed that Artificial Neural Network (ANN) have powerful pattern classification and pattern recognition capacity. They are well suited for prediction problems in which required knowledge is difficult to specify but enough data or observations are available to learn. Although it was proposed in 1964 by Hu [20] brought into use by 1986 when back propagation algorithm was introduced for learning by Rumelhart et al [21]. Originally they were developed to mimic basic biological neural systems particularly the human brain, composed of number of interconnected simple processing elements called neurons which receive input signal from other nodes, process it and produce output signal to other nodes. Before ANN can forecast any desired task, it has to be trained first. Typical case of ANN i.e. Feed Forward Neural Network (FFNN) was used in the current study. In FFNN, information moves in only one direction i.e. forward from input nodes to output nodes through hidden nodes and there are no loops in the network. FFNN is the most practical ANN model and presented in Figure 1. Artificial Neural Network Model (ANN) in Mat Lab Tool was created for the prediction of software maintainability and Multi-Layer Feed-Forward Neural Network Model was selected for learning using empirical data collected in Section-3 and Section-4. The number of hidden neuron selected as 10 for the sample data collected from these five real life applications. Independent variables selected in the current study are nine metrics as described in Table I and Table II and dependent variable is 'Change'. In total, metrics of 864 classes were collected and combined with corresponding 'Changes' made in that class to generate data points for all five applications mentioned in section 3. We partitioned our data into three parts in the ratio of 3:1:1 for training, testing and validation respectively. 60% of the data was used for training i.e. machine learn from the data patterns using specified algorithm, 20% of the data was used for testing where we check how much the predicted values are closer to actual values, 20% of the data was used for validations. Prediction accuracy is calculated by comparing actual value of 'Change' calculated manually in section 4 with predicted values of 'Change'.

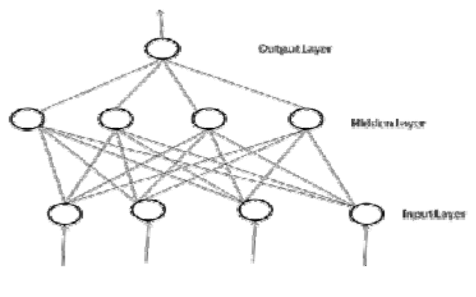

Fig. 1 : A typical Feed Forward Neural Network (FFNN)

VI. DESCRIPTIVE STATISTICS

Descriptive statistics (minimum, maximum, mean, and Median and standard deviation) were calculated and their interpretations are presented in this section. Table-III represents descriptive statistics for class level metrics which were collected for 864 classes consists of five applications FLM, EASY, SMS, IMS and ABP and 233, 292, 129, 96, 114 classes respectively considered in the current study. Following are the observations made from descriptive statistics:

TABLE III : DESCRIPTIVE STATISTICS OF CLASS LEVEL METRICS COLLECTED FOR FIVE SELECTED APPLICATIONS

	FLM System					EASY system				
	Max	Min	Mean	Median	Std Dev	Max	Min	Mean	Median	Std Dev
Cyclomatic Complexity (CC)	29	1	19.31	16	13.76	22	1	20.6	19	14.27
Depth of Inheritance (DIT)	7	1	4.379	5	1.321	5	1	3.6	4	2.503
Coupling Between Object classes	50	3	26.14	30	13.86	54	0	33.5	38.5	21.59
Lines of Code (LOC)	7749	33	189.9	113	207.3	4189	91	207	188	171.7
CCR	5	2	3.276	3	2.978	7	3	4.57	5	5.58
NODBC	12	0	2.483	0	3.532	7	0	2.7	0.5	3.433
WMC (Weighted Methods per Class)	16	1	6.276	5	4.978	23	1	10.5	9.5	8.58
LCOM (Lack of Cohesion of Methods)	0	0	0	0	0	0	0	0	0	0
Maintainability Index (MI)	91	40	61.14	56	18.04	94	43	64.1	56.5	17.92
Change	95	5	41.98	67	45.67	87	9	52.52	63	43.23

	SMS System				
	Max	Min	Mean	Median	Std Dev
Cyclomatic Complexity (CC)	27	1	21.5	19.5	19.62
Depth of Inheritance (DIT)	6	1	3.25	4	2.121
Coupling Between Object classes	59	3	45.38	52.5	18.66
Lines of Code (LOC)	9699	83	367.6	351	219.5
CCR	6	2	4.625	16.5	9.18
NODBC	6	0	3	3	2.507
WMC (Weighted Methods per Class)	29	2	16.63	17.5	9.18
LCOM (Lack of Cohesion of Methods)	0	0	0	0	0
Maintainability Index (MI)	81	49	55.25	52	10.57
Change	79	13	67.89	47	32.43

	IMS System					ABP System				
	Max	Min	Mean	Median	Std Dev	Max	Min	Mean	Median	Std Dev
Cyclomatic Complexity (CC)	13	2	10.79	7	12.39	14	2	10.33	8.5	8.886
Depth of Inheritance (DIT)	5	4	4.029	4	0.171	6	3	4.017	4	0.131
Coupling Between Object classes	30	2	13	13.5	8.09	29	4	14.93	17	8.569
Lines of Code (LOC)	8319	117	43.65	21.5	65.46	7156	171	40.91	32	40.37
CCR	12	0	3.147	3	2.572	11	1	2.483	2	1.847
NODBC	5	0	2.118	1	3.859	8	0	4.931	1	1.041
WMC (Weighted Methods per Class)	12	0	3.147	3	2.572	11	1	2.483	2	1.847
LCOM (Lack of Cohesion of Methods)	3	0	0.147	0	0.558	6	0	0.155	0	0.812
Maintainability Index (MI)	100	48	71.79	67	17.85	100	40	69.5	61	21.04
Change	213	18	79.87	103	67.93	189	19	91.23	78	45.63

- Value of LCOM for FLM, EASY, SMS, IMS and ABP are 0, 0, 0, 3 and 6 respectively which represents that classes are quite cohesive in first three applications.
- Values of DIT for FLM, EASY, SMS, IMS and ABP are 7, 5, 6, 5 and 6 which represents that inheritance is properly exploited in all the selected applications.
- Comments to code ratio is medium in FLM, EASY and SMS and High in IMS and ABP which means IMS and ABP would be easier to understand in maintenance phase.
- Numbers of database connections are more in IMS and ABP systems and less in FLM, EASY and SMS systems.

VII. RESULTS AND DISCUSSION

Many measures have been proposed in literature to estimate the accuracy of different prediction models as part of the studies carried by Conte [22], Kitchenham [23] and Bryson [24], however in the current study we have used most prevalent model i.e. Mean of Absolute Relative Error (MARE) suggested by Kitchenham[23]. To calculate MARE, first we calculate Absolute Relative Error (ARE) using equation (1) followed by Mean of ARE using equation (2). For n observations it is calculated as follows:

$$ARE = \frac{|AV - PV|}{AV} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots.. (1)$$

$$MARE = \frac{1}{n}\sum_{i=1}^{n} AREi \dots\dots\dots\dots\dots\dots\dots. (2)$$

AV is actual values of 'Change' and PV is predicted values of 'Change' using FFNN modeling. MARE values of all five applications FLM System, EASY System, SMS System, IMS System and for ABP System are 0.3478, 0.3676, 0.4769, 0.4966 and 0.4568 as plotted in Figure-2. The prediction accuracy achieved in the current study using

ANN Modeling was compared with other proposed models as determined by Dagpinar et al. [8], Thwin et al [9], Aggarwal et al [10], Koten et al [11], Zhou et al [12], Elish et al [13] and Kaur et al [14] for software maintainability prediction. It can be easily observed that noticeable and quite competitive predication accuracy has been achieved using new proposed metric suite.

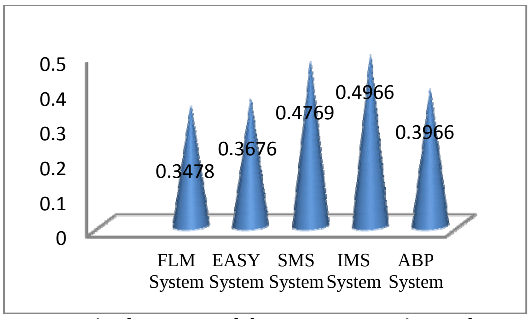

Fig 2 : Applications' and their respective MARE values

To further analyze the results we closely observed the amount of data handled in each application. Maximum values of NODBC for FLM, EASY, SMS, IMS and ABP are 12, 7, 6, 5 and 8 respectively. It is obtained by counting for how many times 'Open ()' is used in each application. We observed that 'EASY' application is more data intensive while IMS is least data intensive when compared against number of database connections made in each application. We also analyzed the results from different perspective to observe the effects of CCR metrics on maintainability which is the ratio of number of code lines to number of comment lines. This metric mainly affects the understandability of the code. The value of CCR metric is highest for FLM system while lowest for SMS system. When the predicted results were compared with actual changes, it was indicated that for FLM systems best prediction accuracy has been achieved and for IMS system, value of MARE is minimum. For FLM systems predictions accuracy was 81% (0.3478/0.4291) more than average prediction accuracy achieved from all other applications. Correlation charts were also drawn between NODBC and 'Change' using SPSS and found to be significant as shown in Figure 3(a). We also observed the correlation between CCR and 'Change' as shown in Figure 3(b). Predicted results are very close to actual values of change. Based on the above findings, followings conclusions are drawn:

- The predicted values are very close to actual values using new metrics suite.
- Proposed metrics (NODBC and CCR) are significantly correlated with software maintainability.
- NODBC and CCR are more impactful while predicting software maintainability for data intensive applications.

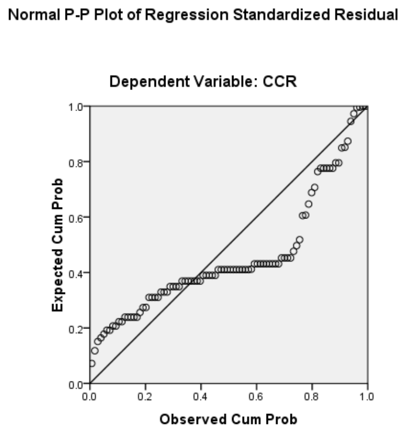

Fig. 3 :(a) Correlation chart between 'NODBC' and 'Change'.

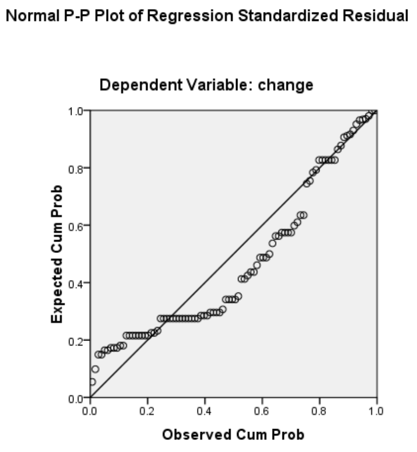

Fig. 3 : (b) Correlation chart between 'CCR' and 'Change'

VIII. THREATS TO VALIDITY

An empirical study always carries few threats to its validity which are equally applicable to the current study and taken into consideration. Empirical data collected from real life application got few specific characteristics and cannot be generalized. Judgments were based on cumulative effects of all metrics on maintainability however it would be more preferable to have controlled environment where at any given time only two new metrics proposed in the study are observed and all other metrics remained constant. Apart from the metrics suite which is used to measure internal quality attributes in the current study, maintainability also depends upon external quality attributes such as quality of the code, expertise of developers, familiarity of the code, reusability etc. In the current study we have considered only internal design metrics and external quality attributes have been deliberately ignored. Although we have tried our best to minimize these threats by taking five different applications which varies greatly on account of database handling, comments and lines of source code. However, these threats can only be removed by conducting more replicated studies using different platforms, different architecture and with different datasets.

IX. CONCLUSION

Objective of our current study was to empirically investigate the effect of newly proposed metrics suite on software maintainability in highly data intensive applications. FFNN modeling techniques was used for the prediction of OO software maintainability using five real life applications. A total of nine important metrics were used to capture attributes of object oriented design and database design. Definition of seven OO design metrics were picked up from the literature and two new metrics were proposed to capture attributes of database design. Newly proposed metrics suite was found to be more meaningful in today's application development scenario which is highly data intensive. Proposed new metrics suite was empirically verified as competitive prediction accuracies were achieved. From the results it can be safely suggested that new metrics suite proposed in the current study appears to be more useful in predicting maintainability of the OO software. Researchers and practitioners developing data intensive applications can make use of this metrics suite for the prediction of software maintainability in early phases of software development in order to achieve better planning of resources. We would like to conduct additional empirical studies in future with an aim to verify and support the findings of this paper on other datasets. Finding of this paper are valid for object oriented medium size system. We are planning to replicate this study for large OO systems. We are also planning to investigate the capability of newly proposed metrics suite in emerging software development scenario such as aspect oriented software development, service oriented software development and component based application development in predicting the maintainability.

REFERENCES

[1] Software Engineering Standards Committee of the IEEE Computer Society, IEEE Std. 828-1998 IEEE Standard for Software Configuration Management Plans, standard, 1998

[2] Aggarwal K. K. and Yogesh Singh, “Software Engineering”, New Age International Publishers, Third Edition, 2008.

[3] Dimitris S, Xenos M, Dimitris C. Relation between Software Metrics and Maintainability. Proceedings of Federation of European Software Measurement Association 1999, Amsterdam, Netherlands: 465-476

[4] Chidamber, Shyam R. and Cris F. Kamerer, “Towards a metrics Suite for Object-Oriented Design Proceedings”, OOPSLA'91, July 1991, pp.197-211 Conference 1991

[5] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability," Journal of Systems and Software, vol. 23, pp. 111-122, 1993.

[6] S. Chidamber and C. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476-493, 1994

[7] W. Li, “Another Metric Suite for Object-oriented Programming”, The Journal of System and Software, 44:155–162,1998

[8] Dagpinar M and Jahnke JH, “Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison”, Proceeding of WCRE '03 Proceedings of the 10th Working Conference on Reverse Engineering; IEEE Computer Society Washington, DC, USA ©2003.

[9] M. Thwin and T. 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.

[10] K. K. Aggarwal, Yogesh Singh, Arvinder Kaur, and Ruchika Malhotra, Application of Artificial Neural Network for Predicting Maintainability using Object- Oriented Metrics, World Academy of Science, Engineering and Technology 22 2006

[11] C. van Koten, A.R. Gray, “An application of Bayesian network for predicting object-oriented software maintainability”, Information and Software Technology, Journal,48(2006)59-67. www.elsevier.com/locate/onfsof

[12] Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines," Journal of Systems and Software, vol. 80, no. 8, pp. 1349-1361, 2007.

[13] M.O.Elish, K.O. Elish, “Application of TreeNet in Predicting OOSoftware Maintainability: A Comparative Study”, European Conference on Software Maintenance and Reengineering , 2009

[14] Arvinder Kaur, Kamaldeep Kaur, Ruchika Malhotra,” Soft Computing Approaches for Prediction of Software Maintenance Effort”, 2010 International Journal of Computer Applications (0975 - 8887) Vol 1,No. 1680

[15] Liang Ping, “A Quantitative Approach to Software Maintainability Prediction “, 2010 International Forum on Information Technology and Applications

[16] Cong Jin, Jin-An Liu , “Applications of Support Vector Machine and Unsupervised Learning for Predicting Maintainability using Object-Oriented Metrics”, 2010 Second International Conference on Multi Media and Information Technology

[17] Ruchika Malhotra and Anuradha Chug,“Software Maintainability Prediction using Machine Learning Algorithms”, Software Engineering: An International Journal (SEIJ). Vol 2 , number 2, pp-19-36, 2012.

[18] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, Empirical Study of Object-Oriented Metrics, Journal of Object Technology, Vol. 5, No. 8, November-December 2006

[19] Zhou Y, Leung H, Xu B, ‘Examining the potentially confounding effect of class size on the associations between object oriented metrics and change proneness’. IEEE Transaction of Software Engineering vol: 35, issue 5 :607–623, 2009

[20] Hu M.CJ. 1964, Application of the adaline system to wether forecasting. master thesis, technical report 6775-i, stanford electronics laboritries, stanford, ca, june

[21] Rumelhart D. E., Hinton GE, Williams RJ: Learning internal Presentation by back-propagating errors, The PDP research Group, Parallel Distributing Processing: Exploration in the Microstructure of cognition, MIT Press, MA, 1994

[22] S. Conte, H. Dunsmore, and V. Shen,” Software Engineering Metrics and Models”. Menlo Park, CA:Benjamin/Cummings, 1986.

[23] B.A. Kitchenham, L.M. Pickard, S.G. MacDonell, M.J. Shepperd,” What accuracy statistics really measure”, IEE Proceedings-Software 148 (3) (2001) 81–85

[24] Bryson AE, Ho YC, “Applied optimal control: optimization, estimation, and control”, Blaisdell Publishing Company or Xerox College Publishing. pp. 481.1969