Maintainability Prediction of Object Oriented Software System by using Artificial Neural Network Approach

Yajnaseni Dash, Sanjay Kumar Dubey, Ajay Rana

Abstract— ***Maintainability is an imperative attribute of software quality. However the prediction of this attribute is a cumbersome process. Therefore various methodologies are proposed so far to estimate the maintainability of software. Among them Artificial Neural Network is one of the sophisticated techniques which have immense prediction capability and this paper explores its application to evaluate maintainability of the object-oriented software. In this study maintenance effort was chosen as the dependent variable and principal components of object-oriented metrics as the independent variables. Prediction of maintainability is performed by Multi Layer Perceptron (MLP) neural network model. The results obtained from the current study are also compared with other models and it is revealed that the presented model is more useful than the previous one.***

Index Terms—Artificial neural network, Maintainability, Object oriented metrics, Principal component analysis.

I. INTRODUCTION

As the object-oriented systems use a large number of small methods, a unique maintenance problem is associated with it. The relationship between OO metrics and software maintenance effort is complex and non linear [1]. Hence this study is conducted on OO paradigm, which is a good platform to assess maintenance effort. In fact, quality estimation means either estimating reliability or maintainability of the software. Maintainability is usually measured as the change in effort i.e. the number of lines changed per class. Artificial Neural network (ANN) is used as a predictive model because of its minimal computation and complex modeling ability. Therefore we established an artificial neural network model to improve prediction outcome of estimating software maintainability.

Manuscript received on April 26, 2012.

Yajnaseni Dash, Department of Computer Science & Engineering, Amity School of Engineering & Technology, Amity University, Sec-125, Noida, U.P., 201301, India, Phone No. +9111204392449 (e-mail: yajnasenidash@gmail.com).

Sanjay Kumar Dubey, Department of Computer Science & Engineering, Amity School of Engineering & Technology, Amity University, Sec-125, Noida, U.P., 201301, India, Phone No. +9111204392449 (e-mail: skdubey1@amity.edu).

Ajay Rana, Department of Computer Science & Engineering, Amity School of Engineering & Technology, Amity University, Sec-125, Noida, U.P., 201301, India, Phone No. +9111204392134 (e-mail: ajay_rana@amity.edu).

II. RELATED WORK

Many researchers described the impact of OO metrics on software maintainability for example, (Rombach [2], Johnson and Foote [3], Moreau [4], Li and Henry [5], Chidamber and Kemerer [6], Basili, Briand and Melo 7], Binkley and Schach [8], Tang, Kao and Chen [9], Muthanna, Kontogiannis, Ponnambalaml and Stacey [10], Genero, M. Piattini, Manso, Cantone [11], Hayes, Patel and Zhao [12], Breesam [13], Dubey and Rana [14] and Dash, Dubey and Rana [1]). OO methodology has been immensely employed in software engineering. As its use increases rapidly, the current applications scenario requires to be improved for further development process. Li and Henry [5] have made a prediction model which integrates ten object oriented metrics. They used statistical technique to establish a strong relationship between metrics and maintenance effort in OO systems. Khoshgaftaar, Allen, Hudepohl and Aud [15] predicted software quality by using the neural networks as a tool. They classified the modules as either fault prone or non fault prone in a large telecommunications system. They had also made a comparison between the ANN model and a non-parametric discriminate model and found that the ANN model had better predictive accuracy than the other one. Fenton and Neil [16] estimated various software defect prediction models by using size and complexity metrics for predicting defects. They compared fault-proneness estimation models and summarized that software quality is a crucial prerequisite in the system development.

Khoshgoftaar, Allen and Xu [17] established a case study of real time avionics software which could predict the testability of each module. They found from experimentation that ANN is an ensuring technique for constructing predictive models. Fioravanti and Nesi [18] have studied over 200 different OO metrics to identify a suitable model for detecting fault-proneness of classes. They found few metrics were appropriate to identify fault-prone classes. Genero, Olivas, Piattini and Romero [19] have presented a set of metrics to measure the structural complexity of UML class diagrams. They also used them to predict maintainability.

Quah and Thwin [20] 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.

Tian and Noore [21] implicated evolutionary neural network modelling for prediction of software cumulative failure time. Thwin and Quah [22] applied neural networks for software quality speculation. OO metrics was considered as independent variable and prediction was done by two neural network models namely ward neural network and general regression neural network (GRNN). They have shown that GRNN network model can predict more accurately than Ward network model. Hu and Zhong [23] proposed a model based on neural network to predict software module risk. The learning vector quantization network used in their study has predicted software quality.

Arvindar, Kamaldeep and Malhotra [24] predicted the software maintenance effort by application of diverse soft omputing techniques. Two commercial software products were taken as dataset and they observed that soft computing techniques are useful for the construction of accurate models to speculate the maintenance effort. In their analysis maintenance effort was chosen as dependent variable and eight OO metrics as independent variable. Ratra, Randhawa, Kaur and Singh [25] compared early prediction of fault prone modules in software design and for this they have applied clustering and neural network techniques. The performance of the two methods were measured based on their accuracy, the mean absolute error and root mean square error values. Their result signified that the performance of neural network approach is much superior to clustering based approach.

Neural networks will be a proficient skill to solve the problems of prediction, classification or control of software systems [27].

III. OBJECT ORIENTED METRICS

The definition of each metrics that used in dataset is described in Table I. Metrics proposed by Li and Henry [5] and Chidamber & Kemerer [6] were taken into consideration in this investigation.

Table I: Metrics Studied

Metric	Definition
DIT	DIT is the maximum depth of the inheritance hierarchy of each class.
NOC	It counts the number of direct children for each class.
MPC	RFC is used to measure the cardinality of the responses of all the classes used for the study.
RFC	This metric measures the static complexity of all methods of a class.
LCOM	It is used to measures the lack of cohesion of a class.
DAC	This metrics indicate the number of messages passed between the objects of the class.
WMC	DAC metric measures the number of abstract data types defined in a class.
NOM	The NOM metric counts the number of local methods of a class.
SIZE1	This metric is used to define the number of properties in the class.
SIZE2	Size2 is a traditional Line of Code (LOC) metrics.

IV. PRINCIPAL COMPONENT ANALYSIS

Principal components (PCs) are linear combinations of the standardized independent variables. In case of the original data are object oriented metrics, the new PC variables are termed as domain metrics. PC analysis is used to maximize the sum of squared loadings of each factor extracted in turn. In PC analysis a new variable (Pi) is constructed, called Principal Component (PC) out of a given set of variables Xj' s(j = 1,2,....,k) [26].

$$P_1 = b_{11} X_1 + b_{12} X_2 + \ldots\ldots + b_{1K} X_K$$

$$P_1 = b_{21} X_1 + b_{22} X_2 + \ldots\ldots + b_{2K} X_K$$

$$.. \quad .. \quad .. \quad .. \quad ..$$

$$P_1 = b_{k1} X_1 + b_{k2} X_2 + \ldots\ldots + b_{kK} X_K$$

The factor b_{ij}s' are called loadings, and are worked out in such a way that the extracted PCs satisfy the following two conditions.

1) PCs are uncorrelated (orthogonal) and
2) The first PC (PC1) has the highest variance; the second PC (PC2) has the next highest variance and so on.

The variables with high loadings facilitate to make out the dimension PC, but this usually requires some degree of interpretation. As the PCs are independent, orthogonal rotation is used. Varimax rotation is the most frequently used in the literature [26].

The sum of squared values of loadings relating to dimension is referred to as eigenvalue. Eigenvalue or latent root is an integral part of PC. The PCs with eigenvalue greater than 1 is usually taken for interpretation [28].

A. Result of principal component analysis

In this study, DIT, NOC, MPC, RFC, LCOM, DAC, WMC, NOM, SIZE1 and SIZE2 are used in UIMS (User Interface System) dataset [5] to produce principal components. The rotated component matrix was extracted from all the metrics by using varimax rotation method. The values above 0.7 are shown in bold in Table II.

These are the only metrics that are employed to interpret the PCs. For each principal component, we also derived its eigenvalue, percentage of variance and percentage of cumulative variance.

The interpretations of PCs are described as follows:
1) PC1: RFC, LCOM, WMC, NOM, and SIZE2 are coupling, cohesion, complexity and size measure metrics. These five metrics were applied for interpreting the PC1.
2) PC2: DAC and SIZE1 are the data and size measure metrics which interpret the PC2.
3) PC3: DIT and NOC are inheritance measure metrics, and were used to interpret the PC3.

Most of the variations in the raw metric data are accounted to these components. Thus the software measuring metrics are transformed in this manner to component or domain measuring metrics. These domain measures are used as the input variables in neural-network modeling [22]. In UIMS system, PCA capture 86.4% of the data set cumulative variance. The detailed results of

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

the principal component analysis are shown in Table II.

The artificial neural network is employed for prediction of maintenance effort. All the domain metrics are used as input to the neural network. If the correlated raw object oriented software metrics used directly then the neural net models will not train properly [15]. The accurate neural training can be done by transferring the raw data into principal components.

Table II: Principal Components

Rotated Component Matrix			
	Principal Components		
Metrics	**PC1**	**PC2**	**PC3**
DIT	-0.017	-0.349	**-0.790**
NOC	0.125	0.021	**0.852**
MPC	0.683	0.337	-0.257
RFC	**0.876**	0.427	0.084
LCOM	**0.884**	0.082	0.129
DAC	0.271	**0.903**	0.240
WMC	**0.938**	0.175	0.161
NOM	**0.752**	0.580	0.195
SIZE1	0.651	**0.710**	0.217
SIZE2	**0.930**	0.269	0.087
Eigenvalues	6.336	1.558	0.745
% of Variance	63.364	15.584	7.452
Cumulative % of Variance	63.364	78.948	86.400

V. MULTI LAYER PERCEPTRON (MLP) NETWORK MODEL

The Neural network is a network of interconnected neurons where information propagation occurs by firing electrical pulses via its connections. During the lifetime of a neuron, the connections or weights need to be adjusted. The neural net consists of one input layer to feed raw data to network, one hidden layer to compute that data and one output layer for getting the computed results. The neural network is characterized by the weighted connections between neurons [27]. The Multi Layer Perceptron network is a function of independent variables which minimize the prediction error of output variables. This network is also called feed forward architecture as it does not contain any feedback loops. The connections flow forward from the input layer to the output layer in the network.

Generally the ANN adjusts the values of the weights to produce a specific target output from a particular input [27]. The MLP network was trained by standard error back propagation algorithm with a momentum of 0.9. The minimum square error is followed as the training stopping criterion.

A. Prediction of maintenance effort

To estimate the maintenance effort of commercial software product, UIMS data [5] was used in this study. The maintenance effort is measured by using the number of lines changed per class. A change in a line could be an addition or a deletion. This dimension is used in the study to evaluate the maintainability of the object-oriented systems. We divided data into training, testing, and production sets using 7:2:1 ratio. The production data set is used to evaluate model performance. We develop a MLP network model and compared our model results with Ward neural network results.

B. Experimental results

The goodness of fit of the model is measured by using the coefficient of correlation(r) and mean absolute error. These statistical measures are shown in Table III. The correlation between the predicted change and the observed change is represented by the coefficient of correlation (r).

Table 3: Experimental Comparison Of Uims System

Validity Tests	**Ward [28]**	**MLP Model**
r-correlation coefficient	0.7798	0.946
Mean absolute error	31.908	17.860
p-Value	0.000	0.000

In UIMS system, r value is 0.7798 in Ward neural network whereas in MLP model it is 0.946. It shows that MLP model is better than Ward neural network model which was proposed by Thwin and Quah. The significance level of a cross validation is indicated by a p value. A universally accepted p value is 0.05. We conclude that the impact of our model prediction is valid and useful.

VI. CONCLUSION

The present study predicted the maintainability of object oriented software by using Multi Layer Perceptron neural network model. This MLP model is found to be more appropriate for the estimation purpose. Thus Artificial Neural Networks have shown their ability to provide an adequate model for predicting maintenance effort. The comprehensive experimental analysis showed that the proposed model is quite valuable for real life applications. We will further explore this research field to minimize the issues associated with software maintainability and object-oriented software system.

REFERENCES

1. Y. Dash, S.K. Dubey and A. Rana, “Maintainability Measurement in Object Oriented Paradigm”, International Journal of Advanced Research in Computer Science (IJARCS), Vol.3, no.2, , April 2012, pp. 207-213.
2. H. D. Rombach, “A controlled experiment on the impact of software structure on maintainability”, Software Engineering, IEEE Transactions on, SE-13(3):344–354, March 1987.
3. R. E. Johnson and B. Foote Designing Reusable Classes. Journal of Object-Oriented Programming. 1988, vol. 1, no. 2, pp. 22-35.
4. D. R Moreau and W. D. Dominick, “Object-Oriented Graphical Information Systems: Research Plan and Evaluation Metrics,” Journal of Systems and Software, vol. 10, 1989, pp. 23-28.
5. W. Li and S. Henry, “Object-Oriented Metrics that Predict Maintainability”, Journal of Systems and Software, vol 23, no.2, 1993, pp.111-122.
6. S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design.” IEEE Trans. Software Eng., vol. 20, no. 6, 1994, pp. 476–493.

7. V. Basili, L. Briand and W. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators", IEEE Transactions on Software Engineering, vol. 22, no.10, 1996, pp. 751-761.
8. Binkley and S. Schach, "Validation of the Coupling Dependency Metric as a risk Predictor", Proceedings in ICSE 98, 1998, pp. 452-455.
9. M.H. Tang, M.H. Kao, and M.H. Chen, "An Empirical Study on Object Oriented Metrics," Proc. Sixth Int'l Software Metrics Symp., 1999, pp. 242-249.
10. S. Muthanna, K. Kontogiannis, K. Ponnambalaml and B. Stacey, "A Maintainability Model for Industrial Software Systems Using Design Level Metrics", In Working Conference on Reverse Engineering (WCRE'00), 2000.
11. M. Genero, M. Piattini, E. Manso, G. Cantone, "Building UML class diagram maintainability prediction models based on early metrics", Proceedings 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry, , IEEE, 2003, pp. 263-275.
12. J.H. Hayes, S.C. Patel and L. Zhao, "A Metrics-Based Software Maintenance Effort Model," Proc. 8th European Conference on Software Maintenance and Reengineering (CSMR'04), 24 – 26 Mar. 2004, IEEE Computer Society, 2004, pp. 254 – 258.
13. K. M. Breesam, "Metrics for Object Oriented design focusing on class Inheritance metrics", 2nd International conference on dependability of computer system IEEE, 2007.
14. S.K. Dubey and A. Rana, "A comprehensive assessment of object oriented software system using metrics approach", International journal of computer science and engineering (IJCSE), 2010, pp. 2726-2730.
15. T.M. Khoshgaftaar, E.D. Allen, J.P. Hudepohl and S.J. Aud "Application of neural networks to software quality modelling of a very large telecommunications system," IEEE Transactions on Neural Networks, Vol. 8, No. 4, 1997, pp. 902--909.
16. N. E. Fenton, and M. Neil, (1999), "A Critique of Software Defect Prediction Models", Bellini, I. Bruno, P. Nesi, D. Rogai, University of Florence, IEEE Trans. Softw. Engineering, vol. 25, Issue no. 5, pp. 675-689.
17. T. M. Khoshgoftaar, E. B. Allen, Z. Xu, "Predicting testability of program modules using a neural network", In Proc. of 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology, pp.57-62, 2000.
18. F. Fioravanti and P. Nesi "A study on fault-proneness detection of object-oriented systems", Fifth European Conference on Software Maintenance and Reengineering, pp. 121 –130, 2001.
19. M. Genero, J. Olivas, M. Piattini and F. Romero ""Using metrics to predict OO information systems maintainability", Proceedings. of the 13th International Conference Advanced Information Systems Engineering, Interlaken, Switzerland, 2001.
20. J. T. S. Quah, M. M. T. Thwin, "Prediction of Software Readiness Using Neural Network", In Proceedings of 1st International Conference on Information Technology & Applications, Bathurst, Australia, 2002, pp. 2312-2316.
21. L. Tian and A. Noore, "Evolutionary neural network modelling for software cumulative fault prediction", Reliability Engineering & system safety, vol. 87, pp. 45-51, 2005.
22. 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.
23. Q. Hu and C. Zhong, "Model of predicting software module risk based on neural network"(in Chinese), Computer Engineering and Applications, Vol.43, No.18, pp.106-110, 2007.
24. A. Kaur, K. Kaur and R. Malhotra, "Soft Computing Approaches for Prediction of Software Maintenance Effort," International Journal of Computer Applications, Vol. 1, no.16, 2010.
25. R. Ratra, N.S. Randhawa, P. Kaur, G. Singh," Early Prediction of Fault Prone Modules using Clustering Based vs. Neural Network Approach in Software Systems," IJECT Vol. 2, Issue 4, Oct . –Dec. 2011
26. 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, pp. 140-144, 2006.
27. Y. Dash and S.K. Dubey, "Quality Prediction in Object Oriented System by Using ANN: A Brief Survey", International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Vol. 2, no. 2, February 2012, ISSN: 2277 128X.
28. M. M. T. Thwin, T. S. Quah, "Application of Neural Networks for Estimating Software Maintainability Using Object-Oriented Metrics", Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering, San Francisco, U.S.A, 2003, pp. 69-73.

AUTHORS PROFILE

Ms. Yajnaseni Dash, is Pursuing M.Tech degree in Computer Science & Engineering from Department of Computer Science and Engineering in Amity University Uttar Pradesh, India (2010 -2012). She received her B.Tech degree in Computer Science & Engineering from from Orissa Engineering College affiliated to Biju Patnaik University of Technology, Odisha, India in 2009.

Sanjay Kumar Dubey, is Assistant Professor in the Department of Computer Science and Engineering in Amity University Uttar Pradesh, India. . He is pursuing his Ph. D. in Computer Science and Engineering from Amity University. He has published more than 30 research papers in reputed National & International Journals. His research area includes Software Engineering and Usability Engineering. He has authored two books for engineering students. He is a member of IET.

Prof. (Dr.) Ajay Rana, is Professor and Director, ATPC, Amity University Uttar Pradesh, India. He is Ph. D. (2005) in Computer Science and Engineering from Uttar Pradesh Technical University, India. His research area includes Software Engineering and Software Testing. He has published more than 55 research papers in reputed National & International Journals. He is the author of several books in the area of computer science. He has received numbers of best papers/case studies medals and prizes for his work. He is a member of CSI and IEEE.