World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:10, 2008

Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics

K. K. Aggarwal, Yogesh Singh, Arvinder Kaur, and Ruchika Malhotra

***Abstract*—**Importance of software quality is increasing leading to development of new sophisticated techniques, which can be used in constructing models for predicting quality attributes. One such technique is Artificial Neural Network (ANN). This paper examined the application of ANN for software quality prediction using Object-Oriented (OO) metrics. Quality estimation includes estimating maintainability of software. The dependent variable in our study was maintenance effort. The independent variables were principal components of eight OO metrics. The results showed that the Mean Absolute Relative Error (MARE) was 0.265 of ANN model. Thus we found that ANN method was useful in constructing software quality model.

***Keywords*—**Software quality, Measurement, Metrics, Artificial neural network, Coupling, Cohesion, Inheritance, Principal component analysis.

I. INTRODUCTION

THERE are several metrics proposed in the literature to capture the quality of OO design and code, for example, (Aggarwal et al. [13]; Briand et al., [14, 15]; Bieman and Kang [7]; Cartwright and Shepperd [17]; Chidamber and Kamerer [21, 22]; Harrison et al. [20]; Henderson-sellers [3]; Hitz and Montazeri [18]; Lake and Cook [2]; Li and Henry [27]; Lee et al. [28] Lorenz and Kidd [19]; Tegarden et al [5]).

These metrics provide ways to evaluate the quality of software and their use in earlier phases of software development can help organizations in assessing large software development quickly, at a low cost. But how do we know which metrics are useful in capturing important quality attributes such as fault-proneness, effort, productivity or amount of maintenance modifications. An empirical study of real systems can provide relevant answers. There have been few empirical studies evaluating the impact of OO metrics on software quality and constructing models that utilize them in predicting quality attributes in the system, such as (Basili et al. [26]; Binkley and Schach [1]; Briand et al [16]; Cartwright and Shepperd [17]; Chidamber and Kamerer [23]; El Emam et al. [9]; Gyimothy et al. [24]; Harrison et al. [20]; Li and Henry [27]; Ping et al. [29]).

Khoshgaftaar at al. [25] introduced the use of the neural networks as a tool for predicting software quality. In [25], they presented a large telecommunications system, classifying modules as fault prone or not fault prone. They compared the ANN model with a non-parametric discriminant model, and found the ANN model had better predictive accuracy. We conduct our study in the OO paradigm. However, since the OO paradigm is different from procedural paradigm, different software design metrics have to be defined and used. We explore the relationship between these design metrics and maintainability effort in this paper. Our ANN model aims to predict OO software quality by estimating the number of lines changed per class.

The paper is organized as follows: Section 2 provides overview of existing studies. Section 3 summarizes the metrics studied and describes sources from which data is collected. Section 4 presents the research methodology followed in this paper. The results of the study are given in section 5. Conclusions of the research are presented in section 6.

II. RELATED WORK

Based on a study of eight medium-sized systems, developed by students Basili et al. [26] found that several of the Chidamber and Kamerer metrics were associated with fault proneness. Briand et al. [18] empirically explored the relationship between OO metrics and the probability of fault detection in system classes. Their results indicated that very accurate prediction models could be derived to predict faulty classes.

Yu et al. [29] chose eight metrics and they examined the relationship between these metrics and the fault-proneness. The subject system was the client side of a large network service management system developed by three professional software engineers. It was written in Java consisting of 123 classes and around 34,000 lines of code. First, they examined the correlation among the metrics and found four highly correlated subsets. Then, they used univariate analysis to find out which metrics could detect faults and which could not.

Manuscript received August 24, 2006

Prof. K.K.Aggarwal is Vice Chancellor of GGS Indraprastha University, Delhi, India (email: kka@ipu.edu)

Prof. Yogesh Singh is with GGS Indraprastha University, Delhi, India (email: ys66@rediffmail.com)

Dr. Arvinder Kaur is with GGS Indraprastha University , Delhi, India (e-mail: arvinderkaurtakkar@yahoo.com.)

Ruchika Malhotra (Corresponding Author phone: 91-011-26431421) is with GGS Indraprastha University, Delhi, India (email: ruchikamalhotra2004@yahoo.com).

TABLE I
METRICS STUDIED

Metric	Definition	Sources
Lack of Cohesion (LCOM)	It counts number of null pairs of methods that do not have common attributes.	[22][11] [12]
Number of Children (NOC)	The NOC is the number of immediate subclasses of a class in a hierarchy.	[22][11] [12]
Depth of Inheritance (DIT)	The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.	[22][11] [12]
Weighted Methods per Class (WMC)	The WMC is a count of sum of complexities of all methods in a class. Consider a class K1, with methods M1,…….. Mn that are defined in the class. Let C1,……….Cn be the complexity of the methods. $$\text{WMC} = \sum_{i=1}^{n} C_i$$	[22][11] [12]
Response for a Class (RFC)	The response set of a class (RFC) is defined as set of methods that can be potentially executed in response to a message received by an object of that class. It is given by RFC=|RS|, where RS, the response set of the class, is given by $$\text{RS} = M_i \cup_{all\ j} \{R_{ij}\}$$	[22][11] [12]
Data Abstraction Coupling (DAC)	Data Abstraction is a technique of creating new data types suited for an application to be programmed. DAC = number of ADTs defined in a class.	[27]
Message Passing Coupling (MPC)	It counts the number of send statements defined in a class.	[27]
Number of Methods per Class (NOM)	It counts number of methods defined in a class.	

Gyimothy et al. [24] empirically validated Chidamber and Kamerer [22] metrics on open source software for fault prediction. They employed regression (linear and logistic regression) and machine learning methods (neural network and decision tree) for model prediction.

Most of these prediction models are built using statistical models. ANN have seen an explosion of interest over the years, and are being successfully applied across a range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. Indeed, anywhere that there are problems of prediction, classification or control, neural networks are being introduced. ANN can be used as a predictive model because it is very sophisticated modeling techniques capable of modeling complex functions.

In [25], Khoshgoftaar et al. presented a case study of real-time avionics software to predict the testability of each module from static measurements of source code. They found that ANN is a promising technique for building predictive models, because they are able to model nonlinear relationships.

Our ANN model aims to predict software quality by estimating the number of lines changed per class.

III. RESEARCH BACKGROUND

In this section we present the summary of metrics studied in this paper (Section 2.1) and empirical data collection (Section 2.2).

A. Dependent and Independent Variables

The continuous dependent variable in our study is maintainability. The goal of our study is to empirically explore the relationship between OO metrics and maintenance effort at the class level. We use ANN to predict maintenance effort per class. The independent variables are principal components from OO metrics chosen for this study. The metrics selected in this study are summarized in Table I.

B. Empirical Data Collection

This investigation is to predict the maintenance effort. The commercial software products UIMS (User Interface System) and QUES (Quality Evaluation System) data are used in this investigation, which is presented in [27]. The maintenance effort is measured by using the number of lines changed per class. A line change could be an addition or a deletion. A change of the content of a line is counted as a deletion followed by an addition. This measurement is used in this study to estimate the maintainability of the OO systems. UIMS system consists of 39 classes and QUES system consists of 71 classes.

IV. SOME COMMON MISTAKES

We used the following methodology in this study:

1. The input metrics were normalized using min-max normalization. Min-max normalization performs a linear transformation on the original data [8]. Suppose that $\min_A$ and $\max_A$ are the minimum and maximum values of an attribute A. It maps value v of A to v' in the range 0 to 1 using the formula:

$$v' = \frac{v - \mathbf{min}_A}{\mathbf{max}_A - \mathbf{min}_A} \qquad (1)$$

2. Perform principal components analysis on the normalized metrics to produce domain metrics.
3. We divided data into training, test and validate sets using 3:1:1 ratio.
4. Develop ANN model based on training and test data sets.
5. Apply the ANN model to validate data set in order to evaluate the accuracy of the model.

A. Principal-Component (or P.C.) Analysis

Many OO metrics have high correlation with each other. P.C analysis transforms raw metrics to variables that are not correlated to each other when the original data are OO metrics, we call the new P.C. variables domain metrics [25].

P.C. analysis is used to maximize the sum of squared loadings of each factor extracted in turn [4]. The P.C. analysis aims at constructing new variable (P_i), called Principal Component (P.C.) out of a given set of variables $X_j's(j = 1,2,....,k)$.

$$\begin{aligned} P_1 &= b_{11}X_1 + b_{12}X_2 + ... + b_{1k}X_k \\ P_2 &= b_{21}X_1 + b_{22}X_2 + ... + b_{2k}X_k \\ &. \quad . \quad . \quad . \quad . \\ P_k &= b_{k1}X_1 + b_{k2}X_2 + ... + b_{kk}X_k. \end{aligned} \qquad (2)$$

All b_{ij}'s called loadings are worked out in such a way that the extracted P.C.s satisfy the following two conditions:

(i) P.C.s are uncorrelated (orthogonal) and
(ii) The first P.C. (P_1) has the highest variance; the second P.C. has the next highest variance so on.

The variables with high loadings help identify the dimension P.C. is capturing but this usually requires some degree of interpretation. In order to identify these variables, and interpret the P.C.s, we consider the rotated components. As the dimensions are independent, orthogonal rotation is used. There are various strategies to perform such rotation. We used the varimax rotation, which is the most frequently used strategy in literature. Eigenvalue or latent root is associated with P.C., when we take the sum of squared values of loadings relating to dimension, then the sum is referred to as eigenvalue. Eigenvalue indicates the relative importance of each dimension for the particular set of variables being analyzed. The P.C.s with eigenvalue greater than 1 is taken for interpretation. Given an n by m matrix of multivariate data, P.C. analysis can reduce the number of columns. In our study n represents the number of classes for which OO metrics have been collected. Using P.C. analysis, the n by m matrix is reduced to n by p matrix (where p<m).

B. ANN Modeling

The network used in this work belongs to Multilayer Feed Forward networks and is referred to as M-H-Q network with M source nodes, H nodes in hidden layer and Q nodes in the output layer [10]. The input nodes are connected to every node of the hidden layer but are not directly connected to the output node. Thus the network does not have any lateral or shortcut connection.

ANN repetitively adjusts different weights so that the difference between desired output from the network and actual output from ANN is minimized. The network learns by finding a vector of connection weights that minimizes the sum of squared errors on the training data set. The summary of ANN used in this study is shown in Table II.

TABLE II
ANN SUMMARY

Architecture	
Layers	3
Input Units	8
Hidden Units	9
Output Units	1
Training	
Transfer Function	Tansig
Algorithm	Back Propagation
Training Function	TrainBR

The ANN was trained by the standard error back propagation algorithm at a learning rate of 0.005, having the minimum square error as the training stopping criterion.

C. Performance Evaluation

In this study the main measure used for evaluating model performance is the Mean Absolute Relative Error (MARE). MARE is the preferred error measure for software measurement researchers and is calculated as follows [6]:

$$\text{MARE} = \left(\sum_{i=1}^{n} \left| \frac{estimate - actual}{actual} \right| \right) \div n \qquad (3)$$

where:
estimate is the network output for each observation
n is the number of observations
to estimate whether models are biased and tend to over or under estimate, the Mean Relative Error (MRE) is calculated as follows[6]:

$$\text{MRE} = \left(\sum_{i=1}^{n} \frac{estimate - actual}{actual} \right) \div n \qquad (4)$$

A large positive MRE would suggest that the model over estimates the number of lines changed per class, whereas a large negative value will indicate the reverse.

V. RESULTS

In this section we present the analysis performed to find the relationship between OO metrics and maintainability effort of the classes.

In this section we present the analysis performed to find the relationship between OO metrics and maintainability effort of the classes.

A. Principal Component Analysis Results

In this section the results of applying P.C. analysis are presented. The P.C. extraction analysis and varimax rotation method is applied on all metrics. The rotated component matrix is given in Table III. Table III shows the relationship between the original OO metrics and he domain metrics. The values above 0.7 (shown in bold in Table III) are the metrics that are used to interpret the PCs. For each PC, we also provide its eigenvalue, variance percent and cumulative percent. The interpretations of PCs are given as follows:

- P1: DAC, LCOM, NOM, RFC and WMC are cohesion, coupling and size metrics. We have size, coupling and cohesion metrics in this dimension. This shows that there are classes with high internal methods (methods defined in the class) and external methods (methods called by the class). This means cohesion and coupling is related to number of methods and attributes in the class.
- P2: MPC is coupling metric that counts number of send statements defined in a class.
- P3: NOC and DIT are inheritance metrics that count number of children and depth of inheritance tree in a class.

TABLE III
ROTATED PRINCIPAL COMPONENTS

P.C.	P1	P2	P3
Eigenvalue	3.74	1.41	1.14
Variance %	46.76	17.64	14.30
Cumulative %	46.76	64.40	78.71
DAC	**0.796**	0.016	0.065
DIT	-0.016	-0.220	**-0.85**
LCOM	**0.820**	-0.057	-0.079
MPC	0.094	**0.937**	0.017
NOC	0.093	-0.445	**0.714**
NOM	**0.967**	-0.017	0.049
RFC	**0.815**	0.509	-0.003
WMC	**0.802**	0.206	0.184

B. ANN Results

We employed ANN technique to predict the maintenance effort of the classes. This method is rarely applied in this area. The inputs to the network were all the domain metrics P1, P2, and P3. The network was trained using the back propagation algorithm. Table II shows the best architecture, which was experimentally determined. The model is trained using training and test data sets and evaluated on validation data set. Table IV shows the MARE, MRE, r and p-value results of ANN model evaluated on validation data. The correlation of the predicted change and the observed change is represented by the coefficient of correlation (r). The significant level of a validation is indicated by a p-value. A commonly accepted p-value is 0.05.

TABLE IV
VALIDATION RESULTS OF ANN MODEL

MARE	0.265
MRE	0.09
r	0.582
p-value	0.004

TABLE V
ANALYSIS OF MODEL EVALUATION ACCURACY

ARE Range	Percent
0-10%	50
11-27%	9.09
28-43%	18.18
>44%	22.72

For validate data sets, the percentage error smaller than 10 percent, 27 percent and 55 percent is shown in Table V. We conclude that impact of prediction is valid in the population.

VI. CONCLUSION

This empirical study presents the prediction of maintenance effort using ANN technique. The independent variables were principal components from eight OO metrics. The results presented above shows that these independent variables appear to be useful in predicting maintenance effort. The ANN model demonstrated that they were able to estimate maintenance effort within 30 percent of the actual maintenance effort in more than 72 percent of the classes in the validate set, and with a MARE of 0.265. Thus ANNs have shown their ability to provide an adequate model for predicting maintenance effort.

The performance of ANN model is to a large degree dependent on the data on which they are trained, and the availability of suitable system data will determine the extent to which maintenance effort models can be developed.

More similar type of studies must be carried out with large data sets to get an accurate measure of performance outside the development population. We plan to replicate our study on large data set and industrial OO software system. We further plan to replicate our study to predict models based on early analysis and design artifacts.

REFERENCES

[1] A.Binkley and S.Schach, "Validation of the Coupling Dependency Metric as a risk Predictor", Proceedings in ICSE 98, 452-455, 1998.

[2] A.Lake, C.Cook, "Use of factor analysis to develop OOP software complexity metrics". Proc. 6th Annual Oregon Workshop on Software Metrics, Silver Falls, Oregon, 1994.

[3] B.Henderson-sellers, "Object-Oriented Metrics, Measures of Complexity", Prentice Hall, 1996.

[4] C.R.Kothari. "Research Methodology. Methods and Techniques", New Age International Limited.

[5] D.Tegarden, S. Sheetz, D.Monarchi, "A Software Complexity Model of Object-Oriented Systems. Decision Support Systems", vol. 13, pp.241-262.

[6] G.Finnie and G. Witting, “AI Tools for Software Development Effort Estimation”, International Conference on Software Engineering: Education and practice, 1996.
[7] J.Bieman, B.Kang, “Cohesion and Reuse in an Object-Oriented System”, Proc. ACM Symp. Software Reusability (SSR’94), pp.259-262, 1995.
[8] J.Han, M. Kamber, “Data Mining: Concepts and Techniques”, Harchort India Private Limited, 2001.
[9] K.El Emam , S.Benlarbi , N.Goel , Rai, “A Validation of Object-Oriented Metrics”, Technical Report ERB-1063, NRC, 1999.
[10] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, “A Neural Net Based Approach to Test Oracle”, ACM SIGSOFT, vol. 29, issue 3, 2004.
[11] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, “Analysis of Object-Oriented Metrics”, International Workshop on Software Measurement (IWSM), 2005.
[12] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, “Empirical Study of Object-Oriented Metrics”, Accepted to be published in Journal of Object-Technology.
[13] K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, “Software Reuse Metrics for Object-Oriented Systems”, Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05), IEEE Computer Society, pp. 48-55, 2005.
[14] L.Briand , W.Daly and J. Wust, “Unified Framework for Cohesion Measurement in Object-Oriented Systems”, Empirical Software Engineering, vol. 3, pp.65-117, 1998.
[15] L.Briand , W.Daly and J. Wust, “A Unified Framework for Coupling Measurement in Object-Oriented Systems. IEEE Transactions on software Engineering”, Vol. 25, pp.91-121, 1999.
[16] L.Briand , W.Daly and J. Wust, “Exploring the relationships between design measures and software quality”, Journal of Systems and Software, Vol. 5, pp.245-273, 2000.
[17] M.Cartwright, M.Shepperd, “An Empirical Investigation of an Object-Oriented Software System”, IEEE Transactions of Software Engineering, 1999.
[18] M.Hitz, B. Montazeri, “Measuring Coupling and Cohesion in Object-Oriented Systems”, Proc. Int. Symposium on Applied Corporate Computing, Monterrey, Mexico, 1995.
[19] M.Lorenz, and J.Kidd, “Object-Oriented Software Metrics”, Prentice-Hall, 1994.
[20] R.Harrison, S.J.Counsell, and R.V.Nithi, “An Evaluation of MOOD set of Object-Oriented Software Metrics”, IEEE Trans. Software Engineering, vol. SE-24, no.6, pp. 491-496, June 1998.
[21] S.Chidamber and C.F.Kamerer, “A metrics Suite for Object-Oriented Design”, IEEE Trans. Software Engineering, vol. SE-20, no.6, 476-493, 1994.
[22] S.Chidamber, C. Kemerer, “Towards a Metrics Suite for Object Oriented design”. Proc. Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA’91). Published in SIGPLAN Notices, vol 26 no. 11, pp.197-211, 1991.
[23] S.Chidamber, D. Darcy, C. Kemerer, “Managerial use of Metrics for Object-Oriented Software: An Exploratory Analysis”, IEEE Transactions on Software Engineering, vol.24 no.8, 629-639, 1998.
[24] T.Gyimothy , R.Ferenc , I.Siket , “Empirical validation of object-oriented metrics on open source software for fault prediction”, IEEE Trans. Software Engineering, vol. 31, Issue 10, pp.897 – 910, Oct. 2005.
[25] T.M.Khoshgaftaar, E.D.Allen, J.P Hudepohl, S.J Aud,., "Application of neural networks to software quality modeling of a very large telecommunications system," IEEE Transactions on Neural Networks, Vol. 8, No. 4, pp. 902--909, 1997.
[26] V.Basili, L.Briand, W.Melo, “A Validation of Object-Oriented Design Metrics as Quality Indicators”, IEEE Transactions on Software Engineering, vol. 22 no.10, pp. 751-761, 1996.
[27] W.Li, S.Henry, “Object-Oriented Metrics that Predict Maintainability”, Journal of Systems and Software, vol 23 no.2, pp.111-122, 1993.
[28] Y.Lee, B.Liang, S.Wu and F.Wang, “Measuring the Coupling and Cohesion of an Object-Oriented program based on Information flow”, 1995.
[29] Yu Ping, Ma Xiaoxing, Lu Jian , “Predicting Fault-Proneness using OO Metrics: An Industrial Case Study”, CSMR 2002, Budapest, Hungary, pp.99-107, 2002.