Int J Syst Assur Eng Manag (Apr-June 2014) 5(2):165–173
DOI 10.1007/s13198-014-0227-4

ORIGINAL ARTICLE

Application of Group Method of Data Handling model for software maintainability prediction using object oriented systems

Ruchika Malhotra · Anuradha Chug

Received: 8 February 2013 / Published online: 20 February 2014

Abstract Object-oriented methodology has emerged as most prominent in software industry for application development. Maintenance phase begins once the product is delivered and by software maintainability we mean the ease with which existing software could be modified during maintenance phase. We can improve and control software maintainability if we can predict it in the early phases of software life cycle using design metrics. Predicting the maintainability of any software has become critical with the increasing importance of software maintenance. Many authors have practiced and proved theoretical validation followed by empirical evaluation using statistical and experimental techniques for evaluating the relevance of any given metrics suite using many models. In this paper, we have presented an empirical study to evaluate the effectiveness of novel technique called Group Method of Data Handling (GMDH) for the prediction of maintainability over other models. Although many metrics have been proposed in the literature, software design metrics suite proposed by Chidamber et al. and revised by Li et al. have been selected for this study. Two web-based customized softwares developed using C# Language have been used for empirical study. Source code of old and new versions for both applications were collected and analysed against modifications made in every class. The changes were counted in terms of number of lines added, deleted or modified in the classes belonging to new version with respect to the classes of old version. Finally values of metrics were combined with "change" in order to generate data points. Hence, in this study an attempt has been made to evaluate and examine the effectiveness of prediction models for the purpose of software maintainability using real life web based projects. Three models using Feed Forward 3-Layer Back Propagation Network (FF3LBPN), General Regression Neural Network (GRNN) and GMDH are developed and performance of GMDH is compared against two others i.e. FF3LBPN and GRNN. With the aid of this empirical analysis, we can safely suggest that software professionals can use OO metric suite to predict the maintainability of software using GMDH technique with least error and best precision in an object oriented paradigm.

Keywords Software maintainability · Group Method of Data Handling (GMDH) · Feed Forward 3-Layer Back Propagation Network (FF3LBPN) · General Regression Neural Network (GRNN) · Empirical validation

1 Introduction

There are many definitions available in literature about 'software maintainability' and more or less all are similar (IEEE 1990; Software Engineering Standards Committee of the IEEE Computer Society 1998; IEEE Standard 1993). IEEE Standard Glossary of Software Engineering Terminology (IEEE 1990) defines 'software maintainability' as the ease with which a software system or component can be

R. Malhotra (✉)
Department of Software Engineering, Delhi Technological University, New Delhi 110042, India
e-mail: ruchikamalhotra2004@yahoo.com

A. Chug
University School of Information and Communication Technology, GGS IP University, Dwarka, New Delhi 110077, India
e-mail: a_chug@yahoo.co.in

modified to correct faults, improve performance or other attributes, or adapt to a changed environment. Data gathered by Capers (2006) about software industry strongly indicates that almost 75 % of project cost is actually spent on maintenance. Maintenance phase starts once the software is delivered to the customer and during this phase changes made into software are inevitable. It not only affects developers in terms of cost and efforts associated but also determines how the customers perceived towards quality as stated by Ash et al. (1994). Cost associated to maintain one line of source code is 10 times the cost of initial development of that line. Hence, software industry today strives hard to find ways and means to measure "Ease of maintainability" not only to reduce cost but also to ascertain whether maintenance of that product is worthwhile or not. As the largest cost incurred for any software product over its lifetime is on account of maintenance (Kaur et al. 2010; Aggarwal et al. 2007; Singh and Goel 2007) and if we really want to control this maintenance costs, we can do so by many ways. One of them would be by utilizing software design metrics during the development phase (Zhou and Leung 2007; Briand et al. 2001; Bandi 2003). Chidamber and Kamerer (1991) and Chidamber and Kemerer (1994) proposed object oriented design metrics as summarized in Table 1 to measure various the features of object orientation such as coupling, cohesion and inheritance etc. Subsequently these were further revised by Li and Henry in year (1993) and two more metrics were added. Since then many researchers Coleman et al. (1994), Bengtsson and Bosch (1999) have used this metric suite on number of applications such as creation of prediction models to establish the relationship of metrics suite with maintainability and empirical validation of various prediction models in terms of better prediction accuracy. Some important studies are Aggarwal et al. (2005b), Aggarwal et al. (2007), Thwin and Quah (2005), Misra (2005), Zhou and Leung (2007), Koten and Gray (2006), Elish and Elish (2009), Jin and Liu (2010), Wang et al. (2009), Dagpinar and Jhanke (2003), Aggarwal et al. (2006), Lucia et al. (2005) and Kaur et al. (2010). These studies have found that there exist a strong link between OO software metrics and its software maintainability. Researchers Heitlager et al. (2007) and Land (2002) have also suggested that in general these design metrics can be used as predictors of "How much would be the maintainability of the product once it is operational?". Contribution of this paper is firstly it presents an empirical validation to determine the relationship between OO metrics and maintainability. Secondly, the paper analyses and compares the performance of Group Method of Data Handling (GMDH) model with two other prediction models, Feed Forward 3-Layer Back Propagation Network (FF3LBPN) and General Regression Neural Network (GRNN). The results are based on the data points collected from two real life web based softwares developed using C# language. After analysing the results it is concluded that GMDH has outperformed and hence it is the best model for the purpose of the prediction of software maintainability.

Table 1 Selection of metrics for study

Metrics	Definition
Weighted methods per class (WMC)	The sum of McCabes's cyclomatic complexities of all local methods in a class. Let a class K1 with method M1…Mn that are defined in the class. Let C1…Cn be the complexity of the methods. $\mathrm{WMC} = \sum_{i=1}^{n} \mathrm{Ci}$
Depth of inheritance tree (DIT)	The depth of a class in the inheritance tree where the root class is zero
Number of children (NOC)	The number of child classes for a class. It counts number of immediate sub classes of a class in a hierarchy
Response for a class (RFC)	The number of local methods plus the number of non local methods called by local methods
Lack of cohesion of methods (LCOM)	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
Message passing coupling (MPC)	The number of messages sent out from a class
Data abstraction coupling (DAC)	The number of instances of another class declared within a class
Number of methods (NOM)	The number of methods in a class
SIZE1 (lines of code)	The number of lines of code excluding comments
SIZE2 (number of properties)	The total count of the number of data attributes and the number of local methods in a class

The rest of this paper is organized as follows. In Sect. 2 we have discussed related studies in the current area. Section 3 describes list of selected software design metrics to measure various aspect of object oriented software and typical application which is studied and analysed on account of selected metrics along with data sets consisting of independent variables and dependent variables. Section 4 presents brief introduction of modelling technique used in the study such as Feed Forward 3-layer Back propagation Algorithm (FF3LBPN), GRNN and GMDH. Section 5 reports the prediction accuracy measures, results and analysis of BPNN, GRNN and GMDH models. Section 6 mentioned about the threats to validity and finally Sect. 7 concludes the paper with outlines and directions for future work.

2 Related work

Many surveys and examples have been presented in past to show that software design metrics are correlated with its

maintainability. Fioravanti and Nesi (2001) presented a prediction model using multi linear regression analysis and validated it on real data. Thwin and Quah (2005) used neural networks to build software quality prediction models. Aggarwal et al. (2005a) 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. Misra (2005) builds maintainability prediction models using Linear Regression techniques. Similarly, in 2006 software maintainability prediction model proposed by Koten and Gray (2006) using Bayesian network was validated for its prediction accuracy. Zhou and Leung (2007) have used multivariate adaptive regression splines (MARS) for predicting object-oriented software maintainability in 2007. They compared the prediction accuracy of proposed model with four other prevailing models: multivariate linear regression (MLR), support vector regression (SVR), artificial neural network (ANN), and regression tree (RT) and stated that MARS is best model to be used as far as maintainability of prediction is concerned. Elish and Elish (2009) proposed the use of TreeNets for the prediction of maintainability. They used the same data set proposed by Chidamber and Kamerer (1991), Li and Henry (1993), and Chidamber and Kemerer (1994) and used by Zhou and Leung (2007), Dagpinar and Jahnke (2003), Thwin and Quah (2005), Koten and Gray (2006), Elish and Elish (2009), and Aggarwal et al. (2006) to compare their results with others. Recently, Malhotra and Chug (2012) have successfully proved the efficiency of GMDH model on the same data set proposed by Li and Henry (1993), Chidamber and Kamerer (1991), and Chidamber and Kemerer (1994) and used by Zhou and Leung (2007), Dagpinar and Jahnke (2003), Thwin and Quah (2005), Koten and Gray (2006), Elish and Elish (2009), and Aggarwal et al. (2006). In their study, it has been proved that GMDH is an excellent method for software maintainability prediction as far as accuracy is concerned. Inspired by the results achieved in Malhotra et al. (2012), the aim of this study is not only to verify whether relationship between metrics and maintainability do exists in real life web based applications but also to verify supremacy of GMDH.

In practice, suitability of selected modelling technique for maintainability prediction not only depends on its ability in capturing the relationships which exists and also the ease with which it could be applied for building prediction models. Accurate software metrics-based maintainability prediction is desirable because firstly it reduces future maintenance efforts by enabling developers to better identify the determinants of software quality and thereby improve design or coding, and secondly, it provides managers with information for more effectively planning the use of valuable resources (Dimitris et al. 1999; Stavironoudis et al. 1999). Although a number of maintainability prediction models have been developed in last decade, they have low prediction accuracies according to the criteria suggested in the literature (Conte et al. 1986; Kitchenham et al. 2001). Therefore, it is necessary to explore new techniques which are easy to use for building maintainability prediction models and should posses' high prediction accuracy. This paper investigates the applicability of novel techniques called GMDH (Ivakhnenko 1966; Ivakhnenko and Koppa 1970; Mohanty et al. 2009) for software maintainability prediction and check if it has better prediction accuracy then existing methods which are based on neural networks namely Feed Forward 3-layer Back Propagation Algorithm (FF3LBPN) and GRNN. In order to do so, the empirical data has been collected and all three prediction models were applied and finally results were compared. We found that GMDH technique is very powerful in its approach as far as prediction of maintainability is concerned. It can do so because during model building, it does not require the specification in advance. It can predict the outcome even with smaller training sets as the computational burden is very less as compared to other models. The procedure used in GMDH automatically filters out input properties that provide little information about location and shape of hyper surface. A multilayer structure maintained in GMDH model is a computationally feasible way to implement multinomial of high degree. Hence, it is very suitable for modelling complex relationships that other modelling techniques find difficult, if not impossible, to reveal. We compare the GMDH model with other prediction models in terms of their prediction accuracy performance. The results suggest that for web based system, GMDH can predict maintainability more accurately than the other modelling techniques.

3 Research background

In this section, we have described the data set used for our study. We first introduced the selection of metrics in the current study, their details and description of dependent and independent variables. We have also described applications which are selected for the study and the empirical data collection process to undertake this study.

3.1 Independent and dependent variables

Several attempts have been made in literature to relate software metrics with software maintainability in object oriented paradigm. Chidamber and Kamerer (1991) presented a OO metric suite to measure various attributes of object oriented paradigm such as inheritance, coupling,

cohesion etc. Since then many proposals of different metrics suite have been given by many researchers. In the last decade many researchers have worked towards empirical validation such as Thwin and Quah (2005), Misra (2005), Zhou and Leung (2007), Koten and Gray (2006), Elish and Elish (2009), Jin and Liu (2010), Wang et al. (2009), Dagpinar and Jhanke (2003), Aggarwal et al. (2006), and Kaur et al. (2010). More or less all have used the metric suite proposed by Chidamber and Kamerer (1991) and Li and Henry (1993). The OO metrics selected in our study are also the same commonly used metric suite and summarized in Table 1.

To determine the object oriented features present in the software we measured the amount of coupling, cohesion and inheritance present in it along with physical aspects such as size. Obviously single metrics is not sufficient to measure such characteristics. Hence the metrics selected and mentioned in Table 1 were considered as independent variables. In order to determine degree of maintainability of software, various prediction models have been proposed, validated and practiced. Primary goal of this paper is to find the effectiveness of GMDH model over other for prediction of maintainability. Dependent variable in our study is "Changes" made into the software during maintenance phase. The changes were counted in terms of number of lines added, deleted or modified in the classes belongs to new version with respect to the classes of old version.

3.2 Empirical data collection

In this paper we have investigated the prediction capability of three models whether they could be used for the prediction of software maintainability using design metrics or not. To empirically validate the effectiveness of models, we have carefully selected two web based software systems namely:

(i) File Letter Monitoring System (FLMS) and
(ii) EASY Classes Online Services collection

Both applications are developed in Microsoft Visual Studio (.NET) software using C# Language. We select these applications and collected metrics to validate the results in this work. FLMS system consists of 55 classes and EASY system consists of 84 classes. Both systems are medium sized software. Source codes of both applications were collected for both versions. Each of the applications was analysed and "Changes" made in second version from the first one were identified. The changes were counted in terms of number of lines added, deleted or modified in the classes belongs to recent version with respect to the classes of first version. Each added or deleted line in source code is considered as one change where as each modification in source code line is counted as two i.e. one addition of new line and one deletion of old line. The careful selection of OO metrics is discussed in Sect. 3.1. Finally values of metrics were combined in order to generate data points to not only examining the effectiveness of GMDH for predicting maintainability but also empirically validating its superiority on real life projects. The data points are divided into 3:1:1 ratio into training, test and validate all three models. The models were used for predicting the value of "Change" based on these generated data points. Finally predicted value received from all three models have compared against actual value to evaluate their prediction accuracy.

4 Research methodology

In this section we have given brief introduction of all three models FF3LBPN, GRNN and GMDH which are used in current study.

4.1 Feed forward 3-layer back propagation algorithm

It is a supervised learning method and extension of ANNs so as to minimize the objective function (Bryson and Ho 1969; Russell and Norvig 2003). It is a multi-stage dynamic system optimization method developed by Bryson and Ho (1969). It is most useful for feed-forward networks which usually have no feedback. The term "Back Propagation" is an abbreviation for "backward propagation of errors". Back propagation requires that the activation function used by the artificial neurons must be differentiable. Every iteration during training starts with extraction of particular data points from training data set. It is then fed through the network in a forward direction and result is produced at the output layer. Now, based on known target information, error is calculated. Necessary changes in weights are determined based upon this error calculation. Changes in weight are calculated layer by layer as a function of the errors determined for all subsequent layers, working backward toward the input layer. This process is repeated until all necessary weight changes are calculated for the entire network. The calculated weight changes are then implemented throughout the network and the next iteration begins. The entire procedure is repeated using the next training pattern.

4.2 General Regression Neural Network (GRNN)

Originally proposed by Specht (1991), GRNN is a modification of Probabilistic Neural Network (PNN) for regression problems. It is a one-pass learning algorithm

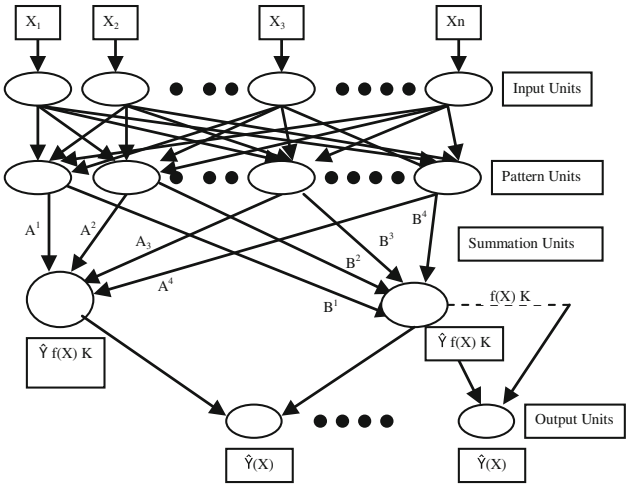

Fig. 1 Architecture of GRNN

with highly parallel structure and provides a smooth transitions from one observed value to another even with sparse data in a multidimensional measurement space (Martin 2011; Rutkowski 2004). If the relationship between independent variables and dependent variables is very complex and not linear in nature, this modified form of regression is a perfect solution. It is very fast in learning and converges to the optimal regression surface as the number of samples becomes very large and allows learning from previous outcomes. The network architecture of a GRNN is similar to that of a PNN except that its summation layer has two neurons that calculate the numerator and denominator. The single neuron in the output layer then performs a division of the two summation neurons to obtain the predicted biological value of the given compound as shown in Fig. 1.

4.3 Group Method of Data Handling (GMDH)

In 1966 it was introduced by Ivakhnenko (1966) and Ivakhnenko and Koppa (1970) for constructing an extremely high order regression type model. The situations where standard multiple regression becomes bogged down in computation and identifying the linear dependence present, this powerful model builds a multinomial of degree in hundreds. The computational burden is quite less in comparison with others and hence it can predict the outcome even with smaller training sets. It can do so because it automatically filters out input properties that provide little information about location and shape of hyper surface. It is based on forward multi-layer neural network structure where learning procedure is self-organized. For the given data, after feeding independent variables as inputs and dependent variables as output, it computes mean square error between actual and

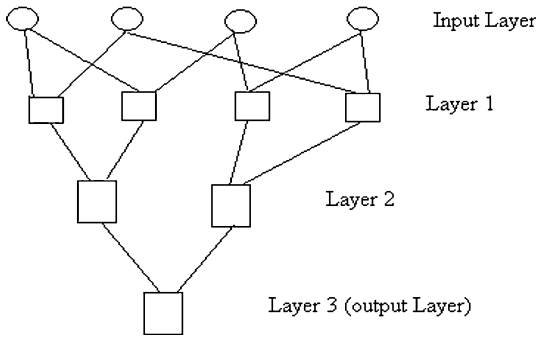

Fig. 2 Architecture of GMDH taken from Malhotra et al. (2012)

predicted value of each unit. Then it sort out the units by MSE value in decreasing order and eliminate bad units i.e. where the error rate is higher. This model keeps adding hidden layers in order to reduce mean square error to minimum. When the MSE become larger than that of the previous layer, stop adding layers and choose the minimum mean square error unit in the highest layer as the final model output. Figure 2 illustrates a typical multi-layer network structure.

5 Results and analysis

In this section we have compared the results of all three prediction models when applied on the data points extracted on real life web based application. In Sect. 5.1 we have discussed various prediction accuracy measured taken from literature and 5.2 onwards, each measure is discussed in detail with corresponding results of each model.

5.1 Prediction accuracy measure

Various measures have been suggested in literature to evaluate the accuracy of any given model (Conte et al. 1986; Kitchenham et al. 2001). Some of the most commonly used measures which have also become de-facto-standards to measure prediction accuracy are MRE, MARE, r-square, R square etc. However, main measure which we have used for evaluating model performance is magnitude of relative error (MARE) proposed by Kitchenham et al. (2001). It is a normalized measure of the discrepancy between actual values and predicted values. It is calculated in two steps. First MRE for each observation is calculated as follows:

$$MRE = | (\textbf{Actualvalue} - \textbf{PredictedValue}) | \div (\textbf{ActualValue})$$

In second step, mean of MRE is calculated as follows:

$$MARE = \frac{\sum_{i=1}^{N} MRE_i}{N}$$

Table 2 Models with their corresponding MARE values

S. no.	Model name	MARE
1	FF3LBPNN	0.4578
2	GRNN	0.5476
3	GMDH	0.3566

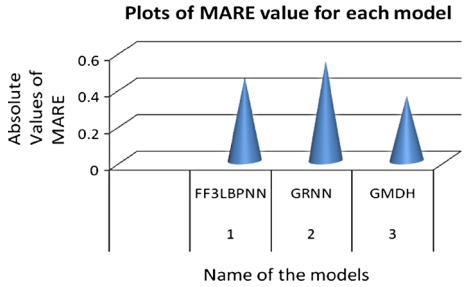

Fig. 3 Plot of MARE value achieved for each model

Table 3 Models and their corresponding pred(0.25) and pred(0.30) values

S. no.	Model name	Pred(0.25)	Pred(0.30)
1	FF3LBPNN	0.51	0.59
2	GRNN	0.44	0.47
3	GMDH	0.61	0.71

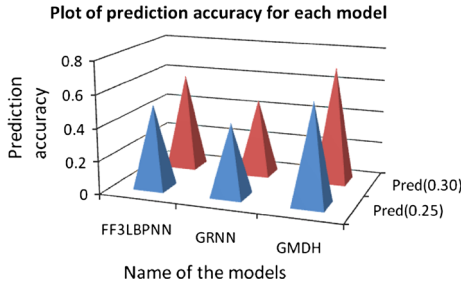

Fig. 4 Plot of pred(0.25) and pred(0.30) value for each model

N is the number of observations. MARE measures the average relative discrepancy. It has the advantages that it is not only independent of measuring unit but also scale independent. In Table 2 we have summarised the MARE values of all three proposed models. The values are plotted in Fig. 3. It can be observed that GMDH has minimum MARE value. We conclude from the results that the impact of GMDH used in the study is valid for the prediction of software maintainability. It also shows that GMDH has good generalization capabilities.

5.2 Prediction accuracy at 25 and 30 %

Fentom and Pfleeger (1997) explains that the prediction accuracy at 25 and 30 % actually measure what proportion of the predicted values have MRE less than or equal to specified value. For example pred(0.25) means how much proportion of the result have MRE less than 0.25 to the total number of observations made. It is given as:

$$Pred(q) = \frac{K}{N}$$

where q is the specified value, K is number of observations whose MRE is less than or equal to q and N is total number of observations. In current study we have taken observations for pred(0.25) and pred(0.30) and summarized them in Table 3. The results are plotted in Fig. 4.

From Fig. 4 it is quite evident that prediction accuracy of GMDH model is much better than other two models not only at 25 % but also at 30 % significantly. At p(0.25) its value is 0.61, which is interpreted as almost 61 % predictions are having less than the error of 25 %. At p(0.30) its value is 0.71 means almost 71 % results are having less than 30 % error in prediction which is quite notable. Values of p(0.25) and p(0.30) for both GRNN and FF3LBPNN have also been observed and found that even though their results are also appreciable but we can safely state that GMDH model has out performed than other two models GRNN and FF3LBPNN as far as prediction accuracy is concerned.

5.3 Percentage of underestimated and overestimated

We have also calculated the percentage of observations that have been underestimated and overestimated by each model. During underestimation we check number of observations where the predicted value is less than actual value and during overestimation we check the number of observations where the predicted value is higher than the actual value. Finally % is calculated as follows:

$$Underestimate = \frac{X}{N} \times 100$$

where X is number of observations where the predicted value is less than the actual value. N is the total number of observations.

$$Overestimate = \frac{Y}{N} \times 100$$

where Y is number of observations where the predicted value is greater than the actual value. N is the total number of observations.

The results are illustrated and plotted in Fig. 5. It has been observed that GRNN tends to slightly overestimate and BPNN model tend to slightly underestimate; but for GMDH, almost 47 % are underestimate and 53 % are

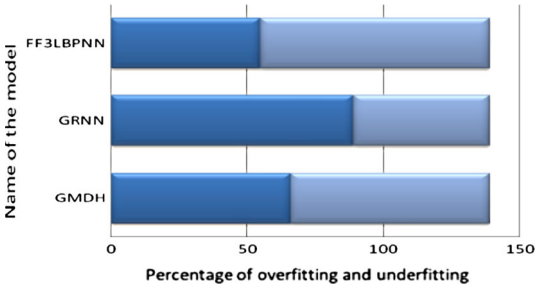

Fig. 5 Percentage of underestimated and overestimated observation by each model

overestimate. This suggests that the GMDH is quite fair in its approach and have tendency neither toward underestimation nor towards overestimation.

6 Threats to validity

There are a few threats to validity of this study that should be taken into consideration while interpreting the results. Some of these limitations are very common whenever any empirical study is conducted and hence not uniquely attributable to this study. However, for the sake of completeness we have mentioned them all as follows:

(1) The results obtained in this study are based on the data obtained from a two real life applications which have specific characteristics and behavior and could not be generalized.
(2) It is exposed here that most of the metrics selected in the study have statistically significant relationship with maintainability. Such statistical methods provide only empirical evidence of relationships and do not demonstrate the actual scenario. Controlled experiments where certain measures can be varied in a controlled manner while keeping the others constant, could actually demonstrate the scenario. As usual however, it is difficult to undertake such controlled experiments in reality.
(3) It should be noted that only code developed in C# was considered in this study. We believed that the conclusions derived from using this code are valid for object-oriented methodologies. However, further research is needed to be conducted to verify this scheme on other object oriented languages such as Java.
(4) There are many psychological factors also which affect maintainability but not considered in this study as we can not quantify them such as different level of expertise for developers, different standards in which application is developed, number of active developers involved, development history of the systems and familiarity of the system with the persons involved in maintenance.
(5) It should be reiterated that this study only provides probable indications of the effect of different design measures on maintainability. Since it was not possible to consider all factors that affect maintainability in one model in this study, the results obtained here are worth verifying as part of future work using the models that consider other factors as well.

7 Conclusions and future work

The life of any software depends on its maintainability. If the software is more maintainable then it is enhance and hence will reduce the maintenance cost significantly for a longer period. This paper presented a study aimed at assessing the efficiency of different prediction models for prediction maintainability of web based systems using Object Oriented metrics. The results show that the GMDH is very helpful model in prediction of software maintainability. After the analysis of results it has been observed that error rate for GMDH model is 35.5 % in comparison with F3LBPNN whose error rate is 45.7 % and GRNN with error rate 54.7 %. Hence, GMDH is found to be more accurate and precise for predicting maintainability of web based applications.

Developers can use this model to judge maintainability of the software while designing and coding. Software practitioners and researchers can use GMDH model in order to predict maintainability in early phases of software development. They can thus reduce maintenance phase and hence save the time. Further they can consider that the developed application is more maintainable or not. This in itself would save the time and cost for the organization responsible for developing and deploying customized software's for the customers to gain their better satisfaction in the industry.

However, it is recommended that further validation studies should be performed using other analytical techniques before the results are actually used in practice. Since the results are based on empirical validation on medium sized web based software more studies needs to be conducted for large systems before generalising the concept. We plan to validate the results on large sized systems in future.

References

Aggarwal KK, Singh Y, Chandra P, Puri M (2005a) Measurement of software maintainability using a fuzzy model. J Comput Sci 1(4):538–542 ISSN 1549-3636

Aggarwal KK, Singh Y, Kaur A, Malhotra R (2005b) Analysis of object-oriented metrics. In: International workshop on software measurement (IWSM)

Aggarwal KK, Singh Y, Kaur A, Malhotra R (2006) Application of artificial neural network for predicting maintainability using object oriented metrics. Proc World Acad Sci Eng Technol 15:285–289

Aggarwal KK, Singh Y, Kaur A, Malhotra R (2007) Application of artificial neural network for predicting fault proneness models. In: International conference on information systems, technology and management (ICISTM 2007), pp 12–13

Ash D, Alderete J, Yao L, Oman PW, Lowther B (1994) Using software maintainability models to track code health. In: Proceedings of international conference on software maintenance. IEEE, Los Alamitos

Bandi R (2003) Predicting maintenance performance using object-oriented design complexity metrics. IEEE Trans Softw Eng 29(1):77–87

Bengtsson P, Bosch J (1999) Architecture level prediction of software maintenance. In: Proceedings of 3rd european conference on software maintenance and reengineering (CSMR '99). IEEE Computer Society, Amsterdam

Briand LC, Bunse C, Daly JW (2001) A controlled experiment for evaluating quality guidelines on the maintainability of object oriented design. IEEE Trans Softw Eng 27(6):513–530. doi:10.1109/32.926174

Bryson AE, Ho YC (1969) Applied optimal control: optimization, estimation, and control. Blaisdell Publishing Company or Xerox College Publishing, Waltham, p 481

Capers J (2006) The economics of software maintenance in twenty first century Version 3, February 14, 2006, Software Productivity Research Inc, research paper. http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf

Chidamber SR, Kemerer CF (1991) Towards a metrics suite for object-oriented design proceedings, OOPSLA'91. ACM Press, Phoenix, pp 197–211

Chidamber S, Kemerer C (1994) A metrics suite for object oriented design. IEEE Trans Softw Eng 20(6):476–493

Coleman D, Ash D, Lowther B, Oman P (1994) Using metrics to evaluate software system maintainability. IEEE Comput 27(8):44–49

Conte S, Dunsmore H, Shen V (1986) Software engineering metrics and models. Benjamin/Cummings, Menlo Park

Dagpinar M, Jahnke JH (2003) Predicting maintainability with object-oriented metrics—an empirical comparison. In: Proceeding of WCRE '03 proceedings of the 10th working conference on reverse engineering. IEEE Computer Society, Washington, DC

Dagpinar M, Jhanke JH (2003) Predicting maintainability with object-oriented metric—an empirical comparison. In: Proceeding of the 10th working conference on reverse engineering (WCRE 03), 2003, vol 155. IEEE Computer Society Washington, DC

De Lucia A, Pompella E, Stefanucci S (2005) Assessing effort estimation models for corrective maintenance through empirical studies. Inf Softw Technol 47(1):3–5

Dimitris S, Xenos M, Dimitris C (1999) Relation between software metrics and maintainability. In: Proceedings of Federation of European Software Measurement Association 1999, Amsterdam, the Netherlands, pp 465–476

Elish MO, Elish KO (2009) Application of TreeNet in predicting object-oriented software maintainability: a comparative study. In: European conference on software maintenance and reengineering, pp 1534–5351. doi:10.1109/CSMR.2009.57

Fentom NE, Pfleeger SL (1997) Software metrics: a rigorous and practical approach, 2nd edn. PSW publishing Company, Boston

Fioravanti F, Nesi P (2001) Estimation and prediction metrics for adaptive maintenance effort of object oriented systems. IEEE Trans Softw Eng 27(12):1062–1084

Heitlager I, Kuipers T, Visser J (2007) A practical model for measuring maintainability. In: 6th International conference on quality of information and communication technology, pp 30–39

IEEE (1990) IEEE Standard Glossary of Software Engineering Terminology, report IEEE Std 610.12-1990. IEEE, New York

IEEE Standard (1993) 1219-1993 IEEE Standard for Software Maintenance. INSPEC Accession Number: 4493167. doi:10.1109/IEEESTD.1993.11557

Ivakhnenko AG (1966) Group method of data handling—a rival of the method of stochastic approximation. Sov Autom Control 13(3):43–71

Ivakhnenko AG, Koppa YU (1970) Regularization of decision functions in the group method of data handling. Sov Autom Control 15(2):28–37

Jin C, Liu JA (2010) Applications of support vector machine and unsupervised learning for predicting maintainability using object-oriented metrics. In: Second international conference on multi media and information technology 2010, vol 10

Kaur A, Kaur K, Malhotra R (2010) Soft computing approaches for prediction of software maintenance effort. Int J Comput Appl 1(16):0975–8887

Kitchenham BA, Pickard LM, MacDonell SG, Shepperd MJ (2001) What accuracy statistics really measure. IEE Proc Softw 148(3):81–85

Koten CV, Gray AR (2006) An application of Bayesian network for predicting object-oriented software maintainability. Inf Softw Technol J 48:59–67

Land R (2002) Measurements of software maintainability. Mälardalen University, Department of Computer Engineering, Vasteras

Li W, Henry S (1993) Object-oriented metrics that predict maintainability. J Syst Softw 23:111–122

Malhotra R, Chug A (2012) Software maintainability prediction using machine learning algorithms. Softw Eng Int J (SEIJ) 2:19–36

Martin CL (2011) Applying a general regression neural network for predicting development effort of short-scale programs. Neural Comput Appl 20:389–401. doi:10.1007/s00521-010-0405-5

Misra S (2005) Modeling design/coding factors that drive maintainability of software systems. Softw Qual J 13(3):297–320

Mohanty R, Ravi V, Patra MR (2009) Software reliability prediction using group method of data handling. In: RSFDGrC rough sets, fuzzy sets, data mining and granular computing, LNAI 5908. Springer, Berlin, pp 344–351

Russell S, Norvig P (2003) Artificial intelligence a modern approach, 2nd edn. Prentice Hall, New Jersey

Rutkowski L (2004) Generalized regression neural networks in time-varying environment. IEEE Trans Neural Netw 15(3):576–596

Singh Y, Goel B (2007) A step towards software preventive maintenance. ACM SIGSOFT Softw Eng 32(4):10. doi:10.1145/1281421.1281432

Software Engineering Standards Committee of the IEEE Computer Society (1998) IEEE Std. 828-1998 IEEE Standard for Software Configuration Management Plans, Standard. http://standards.ieee.org/findstds/standard/828-1998.html

Specht DF (1991) A general regression neural network. IEEE Trans Neural Netw 2(6):568–576

Stavironoudis D, Xenos M, Christodolakis D (1999) Relation between software metrics and maintainability. In: Proceedings of the FESMA99 international conference, Federation of European Software Measurement Association, Amsterdam, the Netherlands, pp. 465–476

Thwin M, Quah T (2005) Application of neural networks for software quality prediction using object oriented metrics. J Syst Softw 76(2):147–156

Wang L-J, Hu X-X, Ning Z-Y, Wen-hua KE (2009) Predicting object-oriented software maintainability using projection pursuit regression. In: The first international conference on information science and engineering (ICISE 2009) 2009, vol 845, p 3827. doi:10.1109/ICISE

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

Ruchika Malhotra

Anuradha Chug