International Journal of Soft Computing 8 (5): 335-339, 2013
ISSN: 1816-9503

Defect Density Based Maintenance Prediction: A Neural Network Based Approach

Arun Sharma and Ankur Bhardwaj
Krishna Institute of Engineering and Technology, Ghaziabad, India

Abstract: With increasing complexity of modern software is also intending to put a serious concern about the increasing cost of maintenance. Thus, to design a measurement tools for software maintainability is of paramount importance. Soft computing based techniques are widely used for the prediction problems. It has been seen that in various engineering problems, these techniques have been proved quite useful. Present research proposes a neural network based approach to predict the software maintainability in early phases of the project development so that corrective measures can be initiated to make it more manageable and maintainable.

Key words: Neural network, maintenance, complexity, defect density, software

INTRODUCTION

Software maintenance is a task that is accomplished by every development group after the delivery of the software. It is a set of activities undertaken on a software system following its release for operational use. Software maintenance is recognized as an important part of software development life cycle.

The cost of software maintenance is rising considerably it has estimated that now a days software maintenance cost is >90% of the total cost of software (Koskinen, 2010) whereas it is 50% two decades before. The lacking or incomplete documentation makes extending a system more difficult and costly (Fig. 1).

Given the massive costs and efforts involved in software maintenance, every company should consider to find the ways to make saving here.

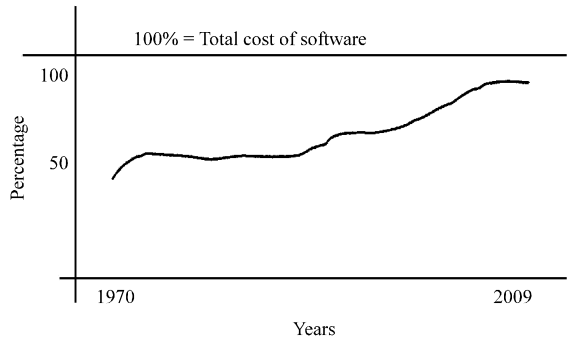

Fig. 1: Development of maintenance cost versus total cost

MAINTAINABILITY

Maintainability is a prediction of the ease with which a system can evolve from its current state to its future desired state. It is termed as the most difficult and costliest activity due to it's inherently involvement in making predictions about the future. Maintainability is defined as IEEE std.:

> The ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to a changed environment

Maintenance can be categorized as: perfective maintenance which is also termed as preventive or enhancement maintenance, adaptive maintenance which concerns external changes and corrective maintenance which refers to modifications initiated by defects in the software.

ESTIMATING MAINTAINABILITY

Maintainability includes concepts of modularity, understandability, changeability, testability, reusability and transferability from one development team to another. These do not take the form of critical issues at the code level. Quantitative measurement of an operational system's maintainability is desirable both as an as a predictor of maintainability over the time and instantaneous measure. Efforts to estimate and track maintainability are intended to help reduce or reverse a

Corresponding Author: Arun Sharma, Krishna Institute of Engineering and Technology, Ghaziabad, India

system's tendency towards degraded integrity and to indicate when it becomes cheaper and/or less risky to rewrite the code than to change it.

Literature review: Quantitative measurement of an operational system's maintainability is desirable both as an instantaneous measure and as a predictor of maintainability over the time. Efforts to estimate and track maintainability are intended to help reduce or reverse a system's tendency towards degraded integrity and to indicate when it becomes cheaper and/or less risky to rewrite the code than to change it.

Hashim and Key (1996) proposed a maintainability model considered two separate qualities: evolvability and reparability. Evolvability involves adaptive and preventive maintenance while reparability involves corrective maintenance. The model describes some factors that affecting maintainability of software components. These factors include readability, modularity, programming language, standardization, level of validation and testing, complexity and traceability.

Another issue of software maintenance of component-based systems by identifying encapsulating and externalizing the variations around design decisions are addressed by Arsanjani *et al*. (2002). The approach is based on the notation of Enterprise Component (EC) which is defined as an architecture patterns that provide a uniform mechanism for management of component boundaries between systems. The process includes the identification of requirements for the system and components, formalizing and abstracting them into a domain-specific language's grammar. This approach enables a highly re-configurable architectural style to help build and maintain reusable components that are responsive and resilient to changing requirements.

Ardimento *et al*. (2004) reports the results of pragmatic study on how characterization of components affects the maintenance effort of the component-based systems and made the assessment that functionality of each component should be as concentrated as possible over a single aspect of the application domain; the training time offered by the component's producer usually indicates the complexity of understanding it and if a component is difficult to understand, then it is also difficult to maintain and a profound knowledge of the component is necessary for the organization before its acceptance, therefore, a trial usage of components is advised before the final decision about their adoption.

Kajko-Mattsson *et al*. (2006) discussed the problems faced by software community towards maintenance and elaborate some concepts for proposing a maintainability model. This model considered both, the product aspects as well as process aspects related with the maintenance.

Wu and Offutt (2003) discussed a new UML-notation based approach for maintaining CBS. They used a static analysis to identify the interface events and dependence relationship that would be affected by the modification in the maintenance activity and also provides a UML-based framework to evaluate the similarities among old and new components.

Singh proposed a new that is based on Artificial Neural Network (ANN) to predict the maintainability of the systems. They considered Readability of Source Code (RSC), Understanding of Software (UOS), Documentation Quality and Average Cyclomatic Complexity (ACC) as independent variables to measure maintainability which is considered as dependent variable. These variables were used to train the ANN by using MATLAB. The results obtained by this method are quite considerable with the prediction quality of 91.42%.

Shukla and Mishra (2008) also used Neural Network based approach to estimate software maintenance efforts. They select 14 factors as cost drivers for study and conducted the experiment by taking various options of number of hidden layers and number of hidden nodes. The input data selected for training was 60% of total while 20% each were used for validation and testing. MRE obtained from the experiment was around 5%. Results concluded that neural network was able to successfully model the maintenance effort.

Aggarwal *et al*. (2006) described the similar approach to predict the maintainability of the Object-Oriented (OO) systems. They considered principal components of eight OO metrics as independent variables. These comprise Lack of Cohesion (LCOM), Number of Children (NOC), Depth of Inheritance (DIT) and others. Maintainability was considered as dependant variable. Results obtained by training the network using back propagation algorithm show that independent variables chosen for the study were able to predict the maintenance efforts with a Mean Absolute Relative Error (MARE) of 0.265.

Aggarwal *et al*. (2005) describes a Fuzzy Model based approach to measure the maintainability of the software system. They considered four factors affecting maintainability these are: average number of live variable, average life span of variables, average cyclomatic complexity and the comments ratio. All inputs are classified into fuzzy sets viz. low, medium and high while

maintainability is classified as very good, good, average, poor and very poor. All inputs and outputs are fuzzified and in total 81 rules that are proposed for the model. Model is validated against the software projects developed by undergraduate engineering students. Yet, the model is not validated on real life complex projects.

Sharma *et al.* (2009) describes that maintainability of component-based system is a measure of five factors mentioned above. These combined factors can be used to estimate the maintainability as it cannot be measured directly. The proposed fuzzy logic based model considers all five factors as inputs and provides a crisp value of maintainability using the Rule Base. All inputs can be classified into fuzzy sets viz. low, medium and high. The output maintainability is classified as very high, high, medium, low and very low. All possible combinations (3^5 i.e., 243) of inputs are considered to design the rule base. Each rule corresponds to one of the five outputs based on the expert opinions.

Riaz *et al.* (2011) describes that the accurate maintainability prediction of relational database-driven software applications can improve the management of projects relating to these applications. The research involved conducting twelve semi-structured interviews with software professionals which were then analyzed using content analysis with the help of NVivo. The results obtain by this method provides the basis for further research involving the proposal and empirical validation of maintainability prediction models for relational database-driven software applications.

Malhotra and Chug (2012) propose few machine learning algorithms with an objective to predict software maintainability and evaluate them. The proposed models are Group Method of Data Handling (GMDH), Genetic Algorithms (GA) and Probabilistic Neural Network (PNN) with Gaussian activation function. The prediction model is constructed using the above said machine learning techniques.

PROPOSED APPROACH FOR PREDICTION OF MAINTENANCE

Software industry is changing very rapidly. So, there is vital need to collect and automate the expertise knowledge. Expertise can help in training the network and after training, it can automate outputs.

In this study, researchers are using the approach of Artificial Neural Network to predict the maintenance of software. Researchers have recognized factors contributing to maintainability and proposed Artificial Neural Network (ANN) Model for evaluating it for software maintenance.

Neural network approach for software maintenance: Neural networks have been established to be an effective tool for pattern classification and clustering (Haykin, 2003). Neural network have been chosen because of the following reason: ANN has pattern classification ability and is adaptive in nature. ANN adjusts the complexity of the network with the complexity of the problem so gives better results than other analytical models.

Artificial neural network: An artificial neural network is characterized by its architecture, learning algorithmic and its activation functions. Learning in the present context may be defined as a change in connection weight values that result in the capture of information that can later be recalled. Generally, the initial weights for the network prior to training are set to random values within a predefined range. Neural network architecture (or network topology) refers to the types of interconnections between neurons. A network is called to be fully connected if the output from a neuron is connected to every other neuron in the next layer. A network with connections that passes outputs in a single direction only to neurons on the next layer is called a feed forward network. A feedback network allows its outputs to be inputs to preceding layers. Feed forward networks are most commonly used for prediction problems as described by Mukherjee and Deshpande (1995). An elementary neuron with R inputs is shown in Fig. 2. Each input is weighted with an appropriate w. The sum of the weighted inputs and the bias forms the input to the transfer function f. Neurons may use any differentiable transfer function f to generate their output.

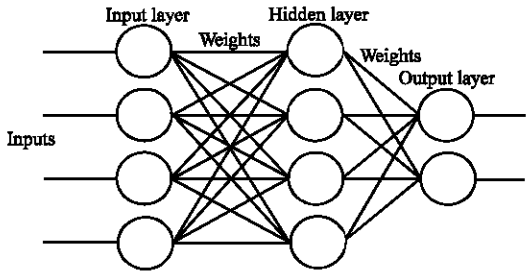

Fig. 2: Neural network architecture

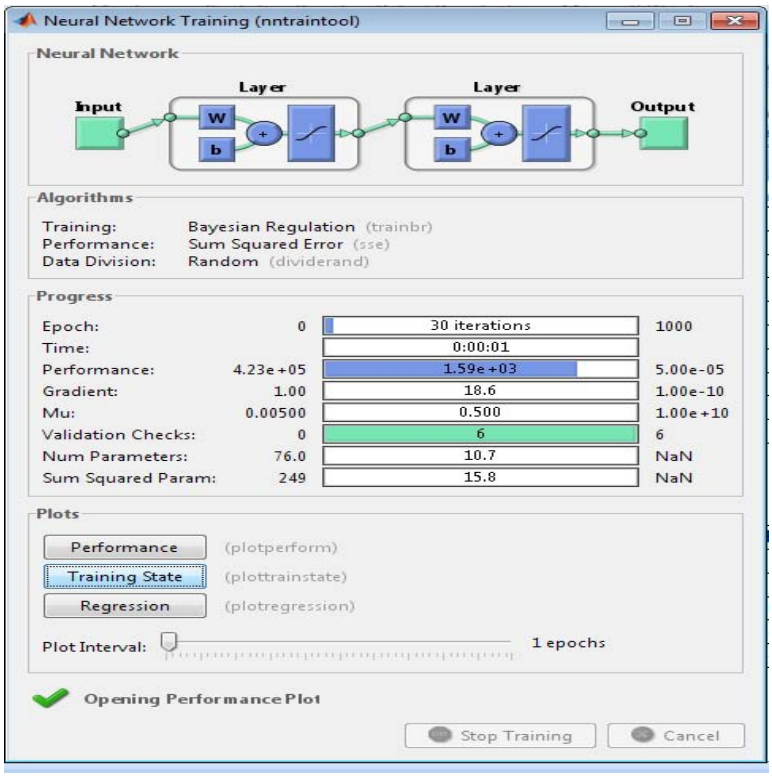

Fig. 3: Snapshot of neural network training tool

Table 1: Parameters values of inputs

Parameters	Values
Training data	830
Testing data	145
Network type	Feed-Forward Backup
Training function	TRAINLM, TRAINBR
Adaptive learning function	LEARNGDM
Performance function	MSE
Number of layers	02
Number of neurons	5,10,15,20
Transfer function	TANSIG
Error	0.00005
Epoch	1000

EXPERIMENTAL IMPLEMENTATION STRATEGIES

In this study, researchers are taking the data from development phases of eclipse, an IDE of Java. This filtered data includes for around 975 files and consists of cyclomatic complexity, pre_release errors and post_release errors. There are total 3 releases of the product, however, researchers have considered only for release-1 of the software release. The various parameters for testing and training of the network is given in Table 1. The snapshot of MATLAB implementation of the experiment is shown in the Fig. 3.

Table 2: Results of parameter TRAINLM and TRAINBR

	TRAINLM	TRAINBR
Number of neurons	RMSE	RMSE
5	0.094092	0.096494
10	0.094035	0.091905
15	0.092612	0.091766
20	0.094717	0.094743

EXPERIMENTAL RESULTS

After applying the various training function with varying the numbers of neurons researchers obtain the Table 2. The experimental results show that the RMSE value for training function TRAINBR for number of neurons 15 at epoch 24 is maximum which indicates that at this level it achieve highest performance.

CONCLUSION

Complexity and defect density are the two main factors affecting the maintainability at later stages. In the present study, researchers proposed a neural network based approach to predict the maintainability.The data from eclipse project have been used for testing and validating the proposed approach. The results obtained

from the experiment are encouraging and leads to the further improving the result by applying other techniques and on some other real time data.

REFERENCES

Aggarwal, K.K., Y. Singh, A. Kaur and R. Malhotra, 2006. Application of artificial neural network configurable architectural object-oriented metrics. Trans. Engin. Comput. Technol., 15: 285-289.

Aggarwal, K.K., Y. Singh, P. Chandra and M. Puri, 2005. Sensitivity analysis of fuzzy and neural network models. ACM SIGSOFT Software Engin. Notes, 30: 1-4.

Ardimento, P., A. Bianchi and G. Visaggio, 2004. Maintenance-oriented selection of software components. Proceedings of the 8th European Conference on Software Maintenance and Reengineering, March 24-26, 2004, IEEE., pp: 115-124.

Arsanjani, A., H. Zedan and J. Alpigini, 2002. Externalizing component manners to achieve greater maintainability through a highly re-configurable architectural style. Proceedings of the International Conference on Software Maintenance, October 3-6, 2002, Montreal, Canada, pp: 628-637.

Hashim, K. and E. Key 1996. A software maintainability attributes model. Malaysian J. Comput. Sci., 9: 92-97.

Haykin, S., 2003. Neural Networks: A Comprehensive Foundation. Prentice Hall, India.

Kajko-Mattsson, M., G. Canfora, D. Chorean, A. van Deursen and T. Ihme *et al.*, 2006. A model of maintainability-suggestion for future research. Proceedings of the International Multi-Conference in Computer Science and Computer Engineering, June 20-22, 2006, China, pp: 436-441.

Koskinen, J., 2010. Software maintenance costs. Information Technology Research Institute, ELTIS-Project, University of Jyvaskyla, Finland.

Malhotra, R. and A. Chug, 2012. Software maintainability prediction using machine learning algorithms software engineering. Int. J., 2: 19-36.

Mukherjee, A. and J.M. Deshpande, 1995. Neural network based expert systems for structural design. Comput. Struct., 54: 367-375.

Riaz, M., E. Mendis and E. Tempero, 2011. Towards predicting maintainability for relational database-driven software applications: Extended evidence from software practitioners. Int. J. Software Engin. Applic., 5: 107-122.

Sharma, A., P. Grover and R. Kumar, 2009. Predicting maintainability of component based system by using fuzzy logic. Communi. Comput. Info. Sci., 40: 581-591.

Shukla, R. and A.K. Mishra, 2008. Estimating software maintenance effort: A neural network approach. Proceedings of the 1st Conference on India Software Engineering Conference, February 19-22, 2008, Hyderabad, India, pp: 107-112.

Wu, Y. and J. Offutt, 2003. Maintaining evolving component-based software with UML. Proceedings of 7th European Conference on Software Maintenance and Reengineering, March 26-28, 2003, Benevento, Italy, pp: 133-142.