Investigation of Software Maintainability Prediction Models

Aida Shafiabady*, Mohd Naz'ri Mahrin*, Masoud Samadi**

* Advanced Informatics School, Universiti Teknologi Malaysia, Jalan Semarak, Kuala Lumpur, Malaysia

**Malaysia-Japan International Institute of Technology, Universiti Teknologi Malaysia, Jalan Semarak, Kuala Lumpur, Malaysia

aishafiabady@gmail.com, mdnazrim@utm.my, solariseir@IEEE.org

Abstract— **Software must be well developed and maintainable to adapt to the constantly changing requirement of the competitive world. In this article, we distinct different software maintainability prediction models and techniques which can help us to predict the maintainability of software, and can lead us to minimum the effort required to fix the faults in the software and the software will be more maintainable. We have gathered our data from different studies focused on the accuracy of the prediction models as criteria. The results of our study showed that there is a little evidence on the accuracy results of the software maintainability prediction models.**

Keywords— **Software maintainability models, software maintenance, techniques, prediction, metrics**

I. INTRODUCTION

Maintainability is a design quality attribute which described as "the ability of the system to undergo changes with a degree of ease, these changes could impact components, features and interfaces when changing the functionality or fixing errors" [1]. Software maintainability is a key quality attributes of the software systems. Maintainability is one of the important quality attributes which can lead in decreasing the cost of the software [2]. There is a belief that a software with high quality is a software that easy to maintain, so the minimum time and effort need to fix the faults and this will lead to decent maintainability and easily maintainable software [3]. Several maintainability prediction models have been proposed so far to help the engineers in assessing the maintainability of software for better development and improvement of the software system.

Prediction which also known as estimation is one of the important part of the planning of the project [4]. Prediction is the act of making educated estimate based on previous information as to the outcome of the results. Although guaranteed accurate prediction results in many cases is impossible but having precise prediction results can assist in making good plans on decrease the effort required to fix the faults of the software, so in this paper we analyse the prediction results of the maintainability of the software from the accuracy perspective point of view.

II. LITERATURE REVIEW

Since now various software maintainability prediction models and techniques were developed. In order to be able to predict the maintainability of software we have to be able to measure the maintainability of software. Several models have been proposed for calculating the maintainability of software. Table1 showed the summarized data of maintainability models which used for measurement intention.

TABLE 1. SUMMARIZE OF MAINTAINABILITY MODELS

No.	Model	Maintainability Measurement
(1)	McCall	modularity, simplicity and conciseness
(2)	Boehm	testability, modifiability and understandability
(3)	Sneed-Mercy	Average number of variables, the Comments Ratio (CR) and the Average Cyclomatic Complexity (ACC)
(4)	Li-Henry	DIT, NOC, RFC, WMC, CBO, LCOM and NOM
(5)	Marcela Genero	Size and structural complexity of UML class diagrams
(6)	Aggarwal	LCOM, DIT, WMC, NOC, RFC, DAC, MPC, NOM and CHANGE Artificial Neural Network
(7)	Koten-Gray	LCOM, LOC, DAC, DIT, MPC, WMC, NOM Bayesian Network
(8)	Zhou-Leung	Multiple regression
(9)	Malhotra Model	Group Method of Data Handling (GMDH)
(10)	Dubey	Multi-Layer Perceptron (MLP) neural network

A. McCall's model

Jim McCall proposed a software quality model in 1977 which called as McCall's model to improve the quality of the software products[5]. In this model, he categorized three main quality attributes of software products that are product

revision, product transition and product operations. Product revision is also defined as the ability to change. Product transition is the adaptableness to the new environments and product transition is basic functioning features. These attributes broken down into eleven quality factors that shows in Figure 1. Each of these quality factors has one or more criteria which defined by McCall as a way to assess the quality of software product. For calculating the maintainability which is one of the quality factors of this model, the criteria used is modularity, simplicity and conciseness [6].

The idea behind this model is that the complete software product picture could be provided by the quality factors identified [7]. The quality metric will have achieved by questions which have relations to each other and the answering will be "yes" or "no". If the answers for "yes" or "no" was equal, then we will accomplish the 50% of that quality criteria measurement. The analysis of the results of this model is based on the judgment of one or more person who answering these questions so the results gathered by this model could be different. The problem existing for this model is that the measurement of the quality factors is difficult because there is no standard or specific method for measuring these factors in this model [8]. With this model the measurement of maintainability done using modularity, simplicity and conciseness.

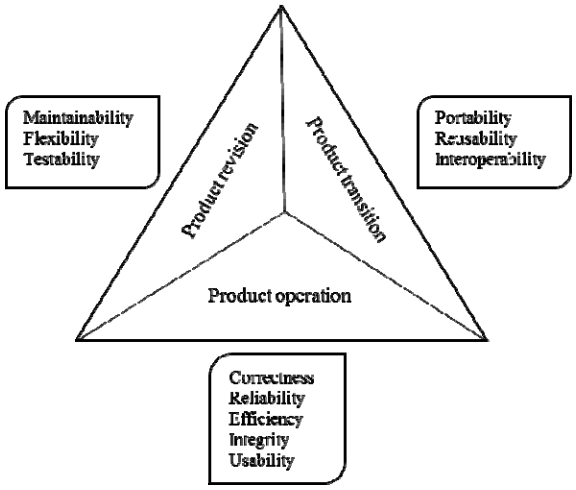

Figure 1. McCall's Quality Factors [9]

B. Boehm Model

This quality model was offered by Barry Boehm in 1978. In this model Boehm added some new factors to McCall's model such as utility and he emphasis on the maintainability of software [10]. In this model in order to evaluate the quality of the software quantitatively, Boehm defined hierarchical model of software product quality. In addition, in his model the evaluation of software product is done with respect to the utility of the program. He defined maintainability as [5] "an ease of identifying what need to be changed." He defined sub-characteristics of maintainability as testability, modifiability and understandability. This model is very difficult to apply in practice but the understand ability of this model is high and it is very easy to learn.

C. Sneed-Mercy Model

Sneed and Mercy proposed a fuzzy model in 1985. This model considered design characteristics [11]. Their model combines three characteristics that is Average number of variables, the Comments Ratio (CR) and the Average Cyclomatic Complexity (ACC) [12]. They also proposed a measurement of maintainability. They defined maintainability as "the measurement of the effort to keep a system operational relative to the effort required to develop it" [11]. They defined some sub characteristics for maintainability such as documentation quality, readability and cohesiveness. According to this model if the CR is low the readability of software will be better therefore the maintainability is better.

D. Li-Henry Model

In 1993 Wei Li and Sallie Henry [13] proposed a model which used the idea of regression model to calculate the maintainability of software. They also defined some software metrics which can be used for maintainability prediction. In 1988 Wake and Henry [14], showed that software maintainability prediction results can be achieved using software metrics. However, so far very few metrics have been proposed, so all those preliminary results led Li and Henry to present software metrics for maintainability prediction. These metrics are DIT, NOC, RFC, WMC, CBO, LCOM, NOM and DAC. Li-Henry metrics are abbreviated in Table 2. These metrics mostly used in object-oriented paradigm and they predict the maintainability of object-oriented software system by measurement of maintenance effort.

TABLE 2. LI-HENRY METRICS

Metric	Definition
DIT	depth of inheritance tree
NOC	number of direct sub classes
RFC	response for classes
WMC	weighted methods per class
CBO	coupling between object classes
LCOM	lack of cohesion in methods
NOM	number of local methods
DAC	data abstraction coupling

E. Marcela Genero Model

Marcela Genero [15] developed a model in 2003 which defined metrics for the size and structural complexity for maintainability measure of UML class diagrams with understand ability time, modifiability time, modifiability completeness, modifiability correctness. In this model, she defined modifiability and understandability as maintainability's sub characteristics. The metrics from this model can be used for software maintainability prediction

resolutions. In order to test her hypothesis, she used Multivariate Linear model, because this model is commonly used, allow relationship between dependent and independent variables counts. The results achieved have an accurate prediction.

F. Aggarwal Model

This model was proposed in 2005 by K.K.Aggarwal and it's an Artificial Neural Network (ANN) based model [16]. In this model, he used Li-Henry metrics to predict the maintainability of software. This model is ANN based model and predict the quality of software by calculating the CHANGE metric that defined as number of lines changes per class. ANN model has a good prediction accuracy results according to the study conducted by Khoshgaftaar at al. [17], so Aggarwal proposed this model in order to have accurate prediction results. This model is trained using backpropagation algorithm and tested data set and the analysis of the results gathered from this model showed that this model is useful in predicting the maintenance effort of the software.

G. Koten-Gray Model

This model was presented in 2006 by Van Koten and Gray. Koten-Gray model [18] is a Bayesian network based maintainability prediction model which can be used for an object-oriented software systems. This model is constructed using Li-Henry model metrics data [13] which were collected from different object-oriented system. In this model they construct a special type of Bayesian networks called Naive Bayes classifier which is a single node representing a classification variable is connected to all other nodes that represent predictor variables. In this model, he used UIMS and QUES datasets to evaluate the proposed model accuracy and the results showed that this Bayesian network based model (Koten-Gray model) have better accuracy comparing with regression analysis based models.

H. Zhou-Leung Model

Yuming Zhou and Hareton Leung [19] employ a regression modeling technique to build a software maintainability prediction model using Li-Henry model metric collected from software systems. They assess the prediction results using UIMS and QUES dataset and they compared their results with other prediction models. In their model for maintainability prediction they used CHANGE metric and count LOC changed during the maintenance period [20].

I. Malhotra Model

This model was presented by Ruchika Malhotra et. al in 2012. They proposed this model to estimate the maintainability of software [21]. They build their prediction model using different machine learning techniques such as Genetic Algorithms (GA) and Group Method of Data Handling (GMDH). They evaluated the performance of this model using UIMS and QUES datasets and they concluded that GMDH network model is one of the best modelling technique to predict the software maintainability.

J. Dubey Model

Maintainability of software can predicted by applying sophisticated techniques such as ANN. So Dubey et. al used Multi-Layer Perceptron (MLP) neural network model to predict maintainability of object oriented software using Li-Henry model metrics such as DIT, LCOM, DAC, NOM [22]. Their proposed model has three input layers and two hidden layers and one output layer. They applied their model to UIMS datasets and the results showed good accuracy.

III. DISCUSSION AND RESULTS

The results of this study shows that there is still no obvious model for predicting the maintainability with good accuracy. Only few of these models support the maintainability with accuracy measurement. McCall's and Boehm's models are traditional models which do not support the accuracy measurement of the maintainability prediction. These models are too complicated to perform and the results are not so reliable.

Investigating these different models shows that Aggarwal Model can predict maintainability having accurate results using ANN. Koten-Gray Model using Bayesian networks can predict maintainability more accurately comparing to regression models. Malhotra model did not consider the accuracy in their prediction model. Dubey model also using ANN can predict maintainability of software with good accuracy. Table 3 shows the classification of different maintainability prediction models.

TABLE 3. TAXONOMY OF DIFFERENT MAINTAINABILITY PREDICTION MODELS

No.	Model	Traditional model	Soft computing based model	Accuracy
(1)	McCall	×		-
(2)	Boehm	×		-
(3)	Sneed-Mercy		×	-
(4)	Li-Henry		×	-
(5)	Marcela Genero		×	-
(6)	Aggarwal		×	Applicable
(7)	Koten-Gray		×	Applicable
(8)	Zhou-Leung		×	Applicable
(9)	Malhotra Model		×	-
(10)	Dubey		×	Applicable

IV. CONCLUSIONS

In this paper we conducted a study on software quality prediction models especially the maintainability of software. This study includes different works done under traditional approaches such as McCall's and Boehm model as well as soft computing models that include machine learning, neural

network and fuzzy models such as Sneed-Mercy model, Aggarwal model. Investigating these different models showed that researchers used different quality characteristics to propose their models like modularity, testability, understand ability. The results gathered from valuing these different models showed that soft computing models based quality prediction achieve better results than traditional models.

V. FUTURE WORK

For future work, we suggest using the combination of soft computing based models and metrics proposed in the past for developing the new maintainability prediction model.

ACKNOWLEDGMENT

We would like to thank the Advanced Informatics School (AIS), Universiti Teknologi Malaysia (UTM) for providing the research facilities.

REFERENCES

[1] Kan, S.H., Metrics and models in software quality engineering. 2002: Addison-Wesley Longman Publishing Co., Inc.

[2] Coleman, D., et al., Using metrics to evaluate software system maintainability. Computer, 1994. 27(8): p. 44-49.

[3] Singh, B. and S.P. Kannojia, A Model for Software Product Quality Prediction. 2012.

[4] Jorgensen, M. and M. Shepperd, A systematic review of software development cost estimation studies. Software Engineering, IEEE Transactions on, 2007. 33(1): p. 33-53.

[5] Saini, R., S.K. Dubey, and A. Rana, Analytical study of maintainability models for quality evaluation. Indian Journal of Computer Science and Engineering, 2011. 2(3): p. 449-454.

[6] Waghmode, M.M.L. and P.P. Jamsandekar, SOFTWARE QUALITY MODELS: A COMPARATIVE STUDY.

[7] Kitchenham, B. and S.L. Pfleeger, Software quality: The elusive target. IEEE software, 1996. 13(1): p. 12-21.

[8] Kokol, P., et al. An analysis of software correctness prediction methods. in Quality Software, 2001. Proceedings. Second Asia-Pacific Conference on. 2001. IEEE.

[9] Milicic, D., Software Quality Attributes and Trade-Offs, chapter Software Quality Models and Philosophies. Blekinge Institute of Technology, 2005: p. 3-19.

[10] Boehm, B.W., J.R. Brown, and H. Kaspar, Characteristics of software quality. 1978.

[11] Sneed, H.M. and A. Mérey, Automated software quality assurance. IEEE Transactions on Software Engineering, 1985. 11(9): p. 909-916.

[12] Bansal, S. and N. Gupta, Software Component Quality Assessment–A Critical Survey. International Research Journal of Computers and Electronics Engineering, 2013. 1(2).

[13] Li, W. and S. Henry, Object-oriented metrics that predict maintainability. Journal of systems and software, 1993. 23(2): p. 111-122.

[14] Wake, S. and S. Henry. A model based on software quality factors which predicts maintainability. in Software Maintenance, 1988., Proceedings of the Conference on. 1988. IEEE.

[15] Genero, M., et al. Building UML class diagram maintainability prediction models based on early metrics. in Software Metrics Symposium, 2003. Proceedings. Ninth International. 2003. IEEE.

[16] Aggarwal, K., et al., Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics. Transactions on Engineering, Computing and Technology, 2006. 15: p. 285-289.

[17] Khoshgoftaar, T.M., et al., Application of neural networks to software quality modeling of a very large telecommunications system. Neural Networks, IEEE Transactions on, 1997. 8(4): p. 902-909.

[18] Van Koten, C. and A. Gray, an application of Bayesian network for predicting object-oriented software maintainability. Information and Software Technology, 2006. 48(1): p. 59-67.

[19] Zhou, Y. and H. Leung, predicting object-oriented software maintainability using multivariate adaptive regression splines. Journal of systems and software, 2007. 80(8): p. 1349-1361.

[20] Mens, T., Serebrenik, A., Cleve, A., Evolving Software Systems. 2014: Springer Science & Business Media.

[21] Malhotra, R. and A. Chug, Software Maintainability Prediction using Machine Learning Algorithms. Software Engineering: An International Journal (SEIJ), 2012. 2(2): p. 19-36.

[22] Dubey, S.K., A. Rana, and Y. Dash, Maintainability prediction of object-oriented software system by multilayer perceptron model. ACM SIGSOFT Software Engineering Notes, 2012. 37(5): p. 1-4.

Aida Shafiabady (M'15), became a Member (M) of IEEE in 2015. She got her BSc degree in software engineering from Islamic Azad University North Tehran Branch in 2008, and her Master degree in software engineering from Universiti Teknologi Malaysia in 2013. Right now, she is a PhD candidate in software engineering at Universiti Teknologi Malaysia. Her research interests includes software quality and assessment, software complexity evaluation, quality prediction, reliability measurement. She is the member of IEEE Computer Society since 2015.

Mohd Naz'ri Mahrin, serves as a Senior Lecturer at the Advanced Informatics School, Universiti Teknologi Malaysia. He received the BSc and MSc degrees in computer science from the Universiti Teknologi Malaysia in 1997 and 2000 respectively. In 2010, he completed his PhD degree in Software Engineering from the University of Queensland, Australia. His research interests include software engineering process and quality, software measurement, usability evaluation, and information security. His current research (as a project leader) are on the effect of varying situations of requirement engineering process in global software development environment and ontology-based software documentation. He is a member of the IEEE Computer Society, ACM Professional, Malaysian Software Engineering Interest Group (MySEIG), and Information Security Professional Association of Malaysia (ISPA).

Masoud Samadi, is a Research officer in Malaysia-Japan International Institute of Technology (MJIIT). He graduated in field of Master of Electrical Engineering from Universiti Teknologi Malaysia in 2014. He received his Bachelor of Hardware engineering in 2007. His research interests lies in Artificial intelligent and Robotics. He is a member of the IEEE from 2009 and ACM Professional since 2013.