IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

Maintainability Evaluation of Object Oriented Software: A Systematic Review

Nidhi Goyal[1], Dr. Reena Srivastava[2]

Ph.D, BBDU, Lucknow India[1]

Dean, School of Computer Applications, BBDU, Lucknow, India [2]

Abstract: Maintainability has always been an elusive concept. Its correct measurement or evaluation is a difficult exercise because of the various potential factors affecting software maintainability. Software maintainability involves external software quality attributes that evaluate the design complexity and effort required for maintaining software. The support provided by software maintainability is significant during development life cycle and quality assurance. The key focus of this review paper is an organized study of maintainability taking into consideration the view provided by its sub factors along with metrics implementation of software maintainability. The aim is to support the maintenance process and facilitate the formation of improved quality software. This paper accomplishes a systematic literature review to study widespread facts of maintainability research, its feature factors and related measurements. A comparative analysis on software maintainability models developed by various researchers/area experts including their contribution and limitation is also presented. In the end our effort is to find the known wide ranging and complete model or framework for quantifying the maintainability of software at an early stage of software development life cycle.

Keywords: Software Maintainability, Maintainability Evaluation, Object Oriented Design, Software Quality, Software testing.

I. INTRODUCTION

Software systems are large, complex and beset with maintenance problems, but at the same time users expect high quality product within time and budget [1]. However, it is tough to evaluate and assure software quality. The ISO/IEC 9126 standard has been developed to address software quality related issues [2]. It describes software product quality characteristics and sub characteristics and proposes metrics for their assessment. It is standard, and might be applied to any type of software product by being tailored to a specific purpose [3]. Quality is the totality of characteristics of an entity that bear on its ability to satisfy stated and implied customers needs [2]. By using the term satisfaction, ISO/IEC 9126 quality model implies the capability of the software to satisfy users in a specified context of use.

The increase in size and complexity of software drastically affects several quality characteristics, specially maintainability and understandability. False interpretation often leads to ambiguities, misunderstanding and hence to faulty software development outcomes. Although the fact that software maintainability and understandability are vital and most considerable components of the software development life cycle, it is poorly managed. This is mainly due to the lack of its proper management and control. Unfortunately, most of the software industries not only fail to deliver a quality oriented software product to their customers, however sometimes they do not understand the relevant quality attributes [11]. Furthermore, in software development industry, schedules are tightly restricted because of the consumer need and pressure; developers are forced to weigh the significance of software quality against the possibility of missing deadlines. For meeting the target, 'on time delivery', testing time is normally reduced. It increases the possibility for defects, leading to problems with the software that include incomplete design, poor quality, high maintenance cost and also the risk of losing customer satisfaction. In order to meet the changing demands of valuable customer or due to many other reasons, software needs to be changed or modified from time to time interval [10].

This procedure of software modification or maintenance is usually carried out by programmers, which may not have developed that software. They need to read and understand source programs and other relevant documents. Even for the software developers of the project, after an interval of few years, it may not be an easy task for them as they themselves might have forgotten the details of the software.

False interpretations can lead to misunderstandings and to faulty development results. Complex design may lead to poor maintainability, which in turn leads to ineffective testing that may result to severe drawbacks and consequence. It is well known fact that flaws of design structure have a strong negative effect on software quality attributes. Other than, creating a high quality design continues to be a poorly defined process [14].Therefore; product design should be developing in such a manner so as to make them easily maintainable and preferably stable.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

II. SOFTWARE MAINTAINABILITY

The key word of maintainability for software first appeared in the categorization of maintenance. It is also planned as the first major attribute of good designed software given by Sommerville [39] in the starting of his book. Maintainability is an essential and precious quality characteristic of software. Software maintainability always supports the maintenance process and assists the creation of superior quality software. An accurate measure of software quality totally depends on maintainability measurement. A lack of maintainability constantly contributes to a higher maintenance charge and effort [7, 8]. The aspiration of increasing the maintainability of object oriented design is not just to detect defects but more significantly, to detect defects as soon as they are introduced [9, 11].

III.CLOSELY RELATED WORK

This section presents the result of a related literature review conducted to collect evidence on object oriented software maintainability evaluation. Broad range of maintainability evaluation models have been proposed in the literature within last two decades. The research on software maintainability first appeared in the year 1975. It is adopted in Jim McCall and Boehm quality model, which build the basis of ISO 9126 software quality model.

Muthanna et al. (2000) developed a maintainability assessment model by the use of polynomial linear regressions. This model was helpful only for procedural software and not suitable for object oriented software. Study highlights that software maintenance is a time consuming and costly phase of a software development life cycle. The authors examined the use of software design metrics to evaluate the maintainability of software systems. A guideline for assessing, estimating and choosing software metrics for predicting software maintainability was presented. In addition, a linear prediction model based on a smallest set of design level software metrics was planned.

Hayes et.al (2003) introduce the Observe Mine Adopt (OMA) model that helps organizations in making improvements to their system development life cycle without committing to and undertaking large scale sweeping industrial process improvement. Especially, the method has been applied to get better software practices focused on maintainability. This novel approach is fully based on the theory that software teams naturally make observations about things that do or do not work well. In the context of software maintainability, it is then essential to perform some measurement to make sure that the method results in enhanced maintainability.

Di Lucca et.al (2004) provided web application based maintainability model faithful to web applications only. Authors stated the increasing distribution of web based services in many and diverse business domains have triggered the need for new web applications. The urgent market demand enforces very short time for the development of new web applications and recurrent modifications for existing ones. The authors introduce a first idea for a web application based maintainability model. Author stated Maintainability of a web based application, with indication to the Source Code Control and Information Structure characteristics may be expressed as a function of the thirty nine (39) attributes: web based application (WA) Maintainability = F (γi, Ai) i=1... 39. Where Ai is the value of 'i^{th}' maintainability attribute & 'γi' is the weight to allocate to that attribute according to how much the attribute affects the maintainability. The proposed model considers those peculiarities that make a web application special from a conventional software system and a set of metrics allowing an assessment of the maintainability is recognized. Results from a few initial case studies to confirm the usefulness of the proposed model are presented in the paper.

Work done by Hayes & Zaho (2005) proposed a maintainability evaluation model that categorized software modules as "easy to maintain" and "not easy to maintain". Such categorization can assist to recognize the modules, which are not easy to maintain. Author performed correlation-analysis and observed that software coding effort correlates with software maintainability. Study developed a new measure that captures the relationship among requirements effort, design effort and coding effort. Next, study built a regression model, namely Maintainability Prediction Model (MainPredMo). Author used regression analysis to construct a prediction model, and obtained the following: MainPredMo =3.795 + 1.652RDCRatio. Analysis showed that this model has a very low statistical significance value of 0.005 and an R square value of 0.64.

Van Koten (2006) presents a Bayesian network maintainability prediction model for object oriented software. The model is developed with the help of object oriented metric data presented in Li and Henry's datasets, which were composed from two dissimilar, object oriented software systems. Prediction correctness of the model is calculated and compared with existing models. This paper evaluates and compares the OO software maintainability prediction model experimentally, using the given maintainability prediction accuracy measures: absolute residual (Ab.Res.), the magnitude of relative error (M.R.E.) and prediction measure. The Ab.Reslt. is the absolute value of residual given by: Ab.Res. =| actual value − predicted value |The results recommend that the Bayesian network model do not calculate maintainability more accurately.

MO. Elish & KO Elish (2009) proposed TreeNet model for maintainability prediction can be consideration of as a series expansion similar to the proper functional relationship. TreeNet model uses two famous object oriented software datasets published by Li and Henry:

UIMS and QUES datasets. The proposed model results designate that competitive maintainability prediction precision has been gained when applying the TreeNet model. Study shows future work would be conducting additional studies with other datasets to extra support the findings of this paper, and to understand the full potential and probable limitation of TreeNet.

Work done by C Jin & JA Liu (2010) presents the applications of support vector machine and unverified learning in object oriented software maintainability prediction via metrics. In this study, the maintainability analysis is carried out at the source code level of development life cycle. The proposed dependent variable was software maintenance effort. Similarly the independent variables were five object oriented metrics determined clustering method. The results showed that the mean absolute relative error was 0.218 of the predictor. Consequently, we found that support vector machine and clustering method were helpful in developing software maintainability predictor. Novel predictor can be used in the related software developed in the same background.

Gautama Kang (2011) highlighted measurement of the software maintainability near the beginning in the development life cycle, particularly at the design time, and it help designers to integrate required improvement and corrections at design phase for improving software maintainability of the delivered software. This paper has proposed a multivariate linear model Compound MEMOOD, which estimates the maintainability of class diagrams of software systems. Earlier MEMOOD model (Maintainability = 0.126 + 0.645 * Understandability + 0.502 *scalability) was developed which estimates the maintainability of the software system on the basis of object oriented metrics. Subsequently study make a comparison of MEMOOD model and Compound MEMOOD model it is found that Compound MEMOOD Model have "R Square" value equals to one which shows that it best fits the data. MEMOOD Model doesn't have R Square value equals 1. Considering the value of R Square author claimed Compound MEMOOD Model has better results than MEMOOD Model. Moreover, no empirical validation has been presented in this study to justify the results.

Alisara Hincheeranan et.al (2012) proposed maintainability estimation tool (MET) consists of the four components. (1) UML Case Tool (2) XMI Parser (3) Metric Calculate (4) Display Results. He stated measuring maintainability of software system at the code level may facilitate a software designer must improves the maintainability of software before deliver to a customer. This work assist a software designer for improves the maintainability of class diagram at code level and facilitate reduces the growing high cost of software maintenance phase. Moreover, no quantitative model has been presented in this study.

Al Dallal, J. (2013) considers classes of three open source software systems. For every class, study accounts for two real maintainability indicators; (1) the number of revised lines of code (2) the number of revisions in which the class was concerned. With 19 internal quality estimates, authors discover the impact of size, cohesion and coupling on class level maintainability. Obtained results show that classes with improved qualities (higher cohesion values and lower coupling and size values) have always better maintainability (i.e. are more possible to be effortlessly modified) than those of inferior qualities. The proposed prediction models can help software designers to find classes with low maintainability.

In the study done by R., & Chug, A. (2014) proposed a new metric suite to overcome the deficiencies and redefine the association among design metrics with software maintainability in data intensive applications. The proposed metric suite is estimated, analyzed using five (5) proprietary software products. The outcomes show that the proposed metric suite is very helpful for maintainability prediction of software systems in general and for data intensive software systems in particular. The proposed metric suite may be considerably useful to the developers in studying the maintainability of intensive software systems before deploying them.

Work done by Singh et al. (2015) focused on a set of object oriented metrics that can be used to evaluate the maintainability of an object oriented design. In this study researcher used the CK metrics to study the effect of various factors related to class and find out which of them have more relevance in measuring the maintainability of software as early as in its design process. During this work author studied metrics, WMC (Weighted Methods per Class), Number of Children (NOC), Depth of Inheritance Tree (DIT), coupling between Objects (CBO) and Response for a Class (RFC) metrics for evaluation of the package designs. Study also found out that value of RFC doesn't need to be low for developing a less fault prone software.

IV. COMPARATIVE STUDY

A complete charting of the existing Maintainability Models Consider by Various Expert has been done in Table 1

After an in depth review, it is apparent that maintainability evaluation should be done at design phase of development life cycle. To evaluate maintainability at design phase it is important to discover maintainability factors that have direct impact on maintainability evaluation. It is obvious from comprehensive literature review that Changeability and Stability is a most important factor for object oriented software maintainability which increases the performance of maintenance process.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

Table 1: A Systematic View of Maintainability Models Consider by Various Expert

Year	Study/Author	Maintainability Evaluation Approach / Model	SDLC Phase	Validation
1984	G.M-Berns	Maintainability Analysis Tool for use with FORTRAN on a VAX	Not given	No Implementation
1985	T.P. Bowens	Average number of days to repair code	Code Level	No
1985	Sneed Mercy	Fuzzy Model	Code Level	No
1987	Kafura and Reddy	Cyclomatic complexity as well as six other software complexity metrics	Code Level	No
1987	Robert Grady (At HP)	FURPS Model	Code Level	Theoretical justification
1991	Geoffrey & kemere	Cyclomatic Complexity Density	Code Level	Yes
1992	Oman Hagemeister	Halstead's Effort (aveE), McCabe' Cyclomatic Complexity (G), LOC (Lines of Code)	Code Level	No
1993	Li Henry	Henry model based on coupling between classes	Code Level	Yes
1994	Coleman Oman	Oman model	Code Level	Yes
1995	Welker Oman	(Improved Oman Model) Cyclomatic Complexity V(g'),LOC (Lines of Code)	Code Level	No
1995	Dromey's "Quality Model"	Quality Model	Code Level	Theoretical justification
2000	Muthanna et al.	Model based on Polynomial Linear Regression	Design Phase	No
2003	Huffman Hayes et al.	Observe Mine Adopt (OMA) Based on Maintainability product	Code Level	No
2004	Lucca Fasolino WAMM	Web Application Maintainability Model	Web based Approach	Web based Approach
2005	Hayes Zaho	(Main Pred Model) LOC (Lines of Code), TCR (True Comment Ratio)	Code level	No
2006	Koten Gray	Bayesian Network Maintainability Prediction Model	Code level	Yes
2008	Prasanth Ganesh & Dalton	With the help of FRT(Fuzzy Repertory Table)	Design Phase	No
2009	MO. Elish & KO Elish	Produced Treenet model using stochastic gradient boosting	Code level	No
2010	C Jin & JA Liu	Based on Support vector machine	Code level	Based on vector machine
2010	S. Rizvi et al.	MEMOOD Model	Design Phase	No
2011	Gautama Kang	Compound Memood Model	Design Phase	No
2012	Alisara et al.	Maintainability Estimation Tool (MET)	Code level	No
2013	Al Dallal, J.	Object-oriented class maintainability prediction using internal quality attributes.	Design and code level	No
2014	R. & Chug A.	A Metric Suite for Predicting Software Maintainability in Data Intensive Applications.	Code level	Based on Metrics
2015	Singh et al.	Estimation of Maintainability in Object Oriented Design Phase: State of the art	Design phase	Theoretical Explanations

DOI 10.17148/IJARCCE.2016.54236

V. CONCLUSION

A lot of maintainability approaches have been proposed in the existing literature for evaluating software maintainability. A review of the related literature shows that most efforts have been put at the later phase of software development life cycle especially at code level. A judgment to change the design in order to get improved maintainability after coding has started is high costly and error prone. For that reason, it is an obvious fact that evaluating maintainability early in the development process greatly reduces maintenance cost, effort, and rework. On the other hand, the lack of maintainability at early stage may not be compensated during subsequent development life cycle. In order to obtain consistent and correct measures of maintainability, it is advisable to recognize the factors that affecting maintainability directly. Though, getting a universally accepted set of maintainability factor is impossible, effort have been made to identify the maintainability major contributors for the same.

REFERENCES

[1] K.K. Aggarwal, Yogesh Singh."Software engineering. " New Age International Publishers, Jan 1, 2005
[2] Hardeep Singh and Aseem Kumar. "A Novel Approach to Enhance the Maintainability of Object Oriented Software Engineering During Component Based Software Engineering. "International Journal of Computer Sci. and Mobile Computing 3.3 (2014): 778-786.
[3] Al Dallal, Jehad. "Object-oriented class maintainability prediction using internal quality attributes." Information and Software Technology 55.11 (2013): 2028-2048.
[4] Pradeep Kumar Singh and Om Prakash Sangwan. "Aspect Oriented Software Metrics Based Maintainability Assessment: Framework and Model." The Next Generation Information Technology Summit (4th International Conference) (2013): 1-07.
[5] McCall, J.A., Richards, P.K., and Walters, G.F., (1977) "Factors in Software Quality", RADC TR-77-369, Vols I, II, III, US Rome Air Development Center Reports.
[6] G. M. Berns. "Assessing software maintainability" .ACM Communications, 27(1), 1984.
[7] Bowen, T. P., Wigle, G. B., Tsai, J. T. 1985. "Specification of software quality attributes ". Tech. Rep. RADC-TR- 85-37, Rome Air Development Center.
[8] Sneed, H., Mercy, A., "Automated Software Quality Assurance". IEEE Trans. Software Eng., (1985)11Bi, 9: 909-916.
[9] Grady, Robert, Caswell, Deborah (1987), "Software Metrics: Establishing a Company-wide Program ". Prentice Hall. pp. p. 159.ISBN 0138218447.
[10] Gill Geoffrey K. and Chris F. Kemerer. (1991). "Cyclomatic Complexity Density and Software Maintenance Productivity", IEEE Transactions on Software Engineering, Dec, pp.1284-1288.
[11] P. Oman and J. Hagemeister, "Metrics for assessing a software system's maintainability, "Software Maintenance, 1992, pp. 337 - 344.
[12] W. Li and S. Henry, "Object-Oriented Metrics that Predict Maintainability", Journal of Systems and Software, vol 23, no.2, 1993, pp.111-122.
[13] D. Coleman, D. Ash, B. Lowther and P. Oman, "Using Metrics to Evaluate Software System Maintainability", IEEE Computer; 27(8), pages 44–49, 1994.
[14] Welker, K. and Oman, P.W., "Software Maintainability Metrics Models in Practice, CrossTalk, Nov./Dec.1995, pp. 19-23 and 32
[15] Geoff R. Dromey's Model," A Model for Software Product Quality", IEEE Transaction on Software Engineering, Feb. 1995, vol. 21 no. 2.
[16] Dromey, R.G., "Concerning the Chimera". IEEE Software 13 (1), pp. 33--43, 1996.
[17] S. Muthanna, K. Kontogiannis, K. Ponnambalaml and B. Stacey, "A Maintainability Model for Industrial Software Systems Using Design Level Metrics", Working Conference on Reverse Engineering (WCRE'00), 2000
[18] M. Genero, M. Piattini, E. Manso, G. Cantone, "Building UML class diagram maintainability prediction models based on early metrics", Proceedings 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry, , IEEE, 2003, pp. 263-275.
[19] Hayes, J. Huffman, Mohamed, N., Gao, T. "The Observe-Mine-Adopt Model: An agile way to enhance software maintain ability", Journal of Software Maintenance and Evolution: Research and Practice, Volume 15, Issue 5, Pages 297 – 323, October 2003.
[20] G. DiLucca, A. Fasolino, P. Tramontana, and C. Visaggio," Towards the definition of a maintainability model for web applications". Proceeding of the 8th European Conference on Software Maintenance and Reengineering, IEEE Computer Society Press, 2004, pages 279– 287.
[21] Kiewkanya, M., Jindasawat, N., Muenchaisri, P., (2004) "A Methodology for Constructing Maintainability Model of Object-Oriented Design," Proc. 4th International Conference on Quality Software, 8 - 9 Sept., 2004, pp. 206 - 213. IEEE Computer Society.
[22] Hayes J.H. and Zaho L (2005), "Maintainability Prediction a Regression Analysis of Measures of Evolving Systems", Proc.21st IEEE International Conference on Software Maintenance, 26-29 Sept.2005, pp.601-604.
[23] C.V. Koten, A.R. Gray, "An application of Bayesian network for predicting object- oriented software maintainability", Information and Software Technology Journal, vol: 48, no: 1, pp 59-67, Jan 2006.
[24] K.K. Aggarwal, Y. Singh, P. Chandra and M. Puri, " Measurement of Software Maintainability Using a Fuzzy Model", Journal of Computer Sciences, vol. 1, no.4, pp. 538-542, 2005 ISSN 1549-3636 © 2005 Science Publications.
[25] K. K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, "Application of Artificial Neural Network for Predicting Maintainability using Object-Oriented Metrics", World Academy of Science, pp. 140-144, 2006.
[26] Subhas Chandra Misra, "Modeling Design/Coding Factors That Drive Maintainability of Software Systems", Software Quality Journal, 13, pages 297- 320, 2005.
[27] Y. Zhou and H. Leung, "Predicting object-oriented software maintainability using multivariate adaptive regression splines", Journal of Systems and Software, vol. 80, no. 8, pp. 1349-1361, 2007
[28] Wang Li-Jin Hu Xin-Xin Ning Zheng-Yuan Ke Wen-Hua ,"Predicting Object-Oriented Software Maintainability Using Projection Pursuit Regression.", Proceedings of the 2005 International Conference on Software Engineering Research and Practice, SERP ,vol.2,pp.942-946.
[29] MO. Elish and KO. Elish, "Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study", European Conference on Software Maintenance and Reengineering, pp 1534-5351, March 2009, DOI 10.1109/CSMR.2009.57.
[30] Rizvi S.W.A. and Khan R.A. (2010) "Maintainability Estimation Model for Object-Oriented Software Design Phase (MEMOOD)", Journal of Computing, Volume 2, Issue 4, April 2010,
[31] Malhotra et.al, "Software Maintainability Prediction using Machine Learning Algorithms." Software Engineering: An International Journal (SEIJ), Vol. 2, No. 2, September 2012
[32] L Ping, "A Quantitative Approach to Software Maintainability Prediction", International Forum on Information Technology and Applications, Vol: 1, No: 1, pp: 105-108, July 2010.
[33] C Jin , A. L. Jin , "Applications of Support Vector Machine and Unsupervised Learning for Predicting Maintainability using Object-Oriented Metrics", Second International Conference on Multi Media and Information Technology , vol 1, no : 1, pp 24-27, April 2010.
[34] Gautam C, kang S.S (2011), "Comparison and Implementation of Compound MEMOOD MODEL and MEMOOD MODEL", International journal of computer science and information technologies, pp 2394-2398.
[35] Malhotra et al. "Software Maintainability Prediction using Machine Learning Algorithms." Software Engineering: An International Journal (SEIJ), Vol.2, No. 2, September 2012 Alisara Hincheeranan

and Wanchai Rivepiboon," A Maintainability Estimation Model and Tool." International Journal of Computer and Communication Engineering, Vol. 1, No. 2, July 2012.

[36] Dubey et.al."Maintainability Prediction of Object Oriented Software System by Using Artificial Neural Network Approach." International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-2, May 2012.

[37] Laxmi Shanker Maurya et.al," Maintainability assessment of web based application.'', Journal of Global Research in Computer Science, Vol 3, No. 7, July 2012.

[38] Sommerville. "Software Engineering". 4th ed. New York, Addison-Wesley, (1992).

[39] McCall, J.A., Richards, P.K., and Walters, G.F., "Factors in Software Quality", RADC TR-77-369, Vols I, II, III, US Rome Air Development Center Reports, (1977)

[40] Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M., (1978) Characteristics of Software Quality, North Holland.

[41] ISO 9126-1 Software Engineering - Product Quality - Part 1: Quality Model, 2001.

[42] Grady, Robert, Caswell, Deborah (1987), Software Metrics: Establishing a Company- wide Program. Prentice Hall. pp. p. 159. ISBN 0138218447.

[43] Sneed, H., Mercy, A. "Automated Software Quality Assurance". IEEE Trans. Software Eng., 11Bi, 9: 909-916, (1985),

[44] Rizvi, S.W.A., Khan, R.A.," Maintainability Estimation Model for Object-Oriented Software in Design Phase", Journal of Computing, Volume 2, Issue 4, (April 2010),

[45] Hordijk, Wiebe, and Roel Wieringa. "Surveying the factors that influence maintainability: research design." ACM SIGSOFT Software Engineering Notes. Vol. 30. No. 5. ACM, 2005.

[46] Vivek Rai, Akhilesh Mohan Srivastava, Himanshu Pandey, Dr. V. K Singh "Estimation of Maintainability in Object Oriented Design Phase: State of the art", International Journal of Scientific & Engineering Research, Volume 6, Issue 9, September-2015.

DOI 10.17148/IJARCCE.2016.54236