Volume 4, Issue 12, December 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Maintainability Quantification of Object Oriented Design: A Revisit

Rajendra Kumar, Namrata Dhanda
Department of Computer Science GITM,
Lucknow, U.P., India

Abstract— Maintainability is one of the most considerable quality indicators. Its truthful quantification always support and enhance the maintenance process. An exact measure of software quality fully depends on maintainability quantification. It is very hard to find a comprehensible view on all the probable factors that have positive effect on software testability quantification. The most significant apprehension of this review paper will be organized study of software maintainability quantification of object oriented design taking into consideration by its factors and metrics implementation , keeping in mind to supports the maintenance process and make possible the formation of improved quality software. At this paper we will discuss regarding some of the important theories specified by several researchers in their paper and we will relate those all research through our paper in order to promote our effort.

Keywords— Software Maintainability, Maintainability Quantification, Object Oriented Design, Software Quality, Software Maintenance

I. INTRODUCTION

Software maintenance is defined as "the course of action of modifying a software organization or module after release to correct errors, progress performance or added features, or adjust to a changed background. Despite the fact that Software maintainability is defined as "the easiness with which a software organization or module can be modified to correct errors, progress performance or added features, or adjust to a changed background [1, 12]. Maintainability is one of the most important characteristics of software quality. The bulk of software companies splurge 60 to 70 percent of funds for correcting, adopting and maintaining the existing software [2].Maintainability quantification facilitates to observe the maintenance effort and easiness of software maintenance at design level. In the early hours evaluation of maintainability assists to make use of its attributes more capably to improve the quality of software. The design phase assessment of software maintainability is more realistic for software development and maintenance cost-effectively [3]. Maintainability is the capability of the system to go through changes with a degree of easiness. Improving system maintainability can enlarge availability and decrease the effects of run-time defects. Software maintenance has need of added effort than any other activity of development life cycle. The maintainability of software is not achievable directly although with the assist of their internal characteristics quantification.

II. SOFTWARE MAINTAINABILITY QUANTIFICATION

Maintainability is one of the most noteworthy characteristics of software quality. The bulk of software companies splurge 60 to 70 percent of resources for correcting, adopting and maintaining the existing software [1]. The majority of companies pay out over 70 percent cost of total development budget on testing, maintenance to manage the software quality [2].Maintainability quantification helps to analyze the maintenance effort and easiness of software at design level. Maintainability quantification in the early hours at design phase is highly emphasized in my proposed study; hence, considered important for the delivery of quality software. Our main passion is that it is for the duration of the analysis and design phase that maintainability analysis can yield the highest payoff. There is a universal agreement in the middle of industry professionals and academicians to link together maintainability with the development life cycle in order to deliver protected, secure and consistent software within time and financial plan [5].

III. MAINTAINABILITY QUANTIFICATION AT DESIGN PHASE

Early assessment of maintainability, absolutely at design phase helps designers to advance their designs earlier than the coding starts. As a result, it appreciably reduces rework throughout and after implementation, as well as facilitates to design useful test plans, improved project and resource planning [4]. A resolution to change the design in order to get better maintainability after coding has started may be very costly and error-prone. At the same time as quantifying maintainability near the beginning in the development process (namely at design phase) may significantly decrease the overall development cost and time and facilitate to produce superior quality maintainable software. Unhappy to say, the majority of the software companies not just fails to produce quality software to their customers, however in addition do not understand the proper quality attributes [1, 25, 14].

The objective of increasing the maintainability of software is not just to detect defects but extra essentially, to detect defects as soon as they are introduced. In consequence, reducing the cost and time to fix the bug and producing higher quality maintainable software each build of the release [13]. Maintainability factor contribute to 40-70% of the cost of software products. Enhanced maintainability at all times direct to reduced maintenance efforts and reduced price and time. After the above conversation our conclusion is that maintainability is a quality factor that attempts to predict how much effort will be required for software maintaining and to quantify the complexity of causing a fault to result in a failure.

IV. CLOSELY RELATED WORKS

Large collection of maintainability quantification models has been deliberated in the research papers within previous two decades. A number of maintainability models were planned to assist the designers in quantifying the maintainability of object oriented software so as to produce superior and enhanced software systems. Opening from 1970s to 2014 a range of maintainability quantification models or techniques was developed. In 1977, Jim McCall considered a software quality model called as McCall's model. In this model McCall acknowledged the 11 quality factors broken down by the three key angles for characterizing the quality attributes of a software product specifically product revision (capacity to change), product transition (adaptability to new environments) and product operations (fundamental operational characteristics).

The maintainability of software is affected by a lot of factors, such as the availability of qualified software staff, the easiness of system management, the use of consistent programming languages etc. [7]. Inadvertent be short of care in design, implementation and testing has a logical negative impact on the capability to maintain the resultant software [8].

On or after the review of literature it has been observed that a variety of researchers planned many models for maintainability assessment, but in almost all of these revisions, maintainability assessment based on the procedures taken later than the coding phase of development life cycle. For the cause that of this, maintainability predictions are ready in the later stages of development life cycle, and it turn out to be extremely difficult, tough and expensive to get better the maintainability at that stage.

An important Look of Maintainability Models Consider by Various Expert

Author/ Approach	Year	Maintainability Quantification Approach	Limitations
McCall's Approach	1977	Simplicity, conciseness, and modularity	Not best suited to maintainability quantification at design phase
Bohem Approach	1978	Understandability ,Modifiability ,Testability	Applicable only later phases of development life cycle.
Berns Approach	1984	Maintainability Analysis Tool for make use of by FORTRAN on a VAX	Method was very difficult and has high Complexity.
Bowen Approach giving Corrective maintainability	1985	average number of days to repair code	Unfortunately all above maintainability metrics measure only higher level of abstraction.
Sneed- Mercy Approach(Fuzzy Approach)	1985	middling number of variables, the middling Cyclomatic Complexity (ACC) and the Comments Ratio (CR)	Maintainability Prediction Model apply at later stage of SDLC
Kafura and Reddy Approach	1987	Based on cyclomatic complexity as well as six other software complexity metrics	This model has various limitations from implementation point of view.
Robert Grady Approach (At HP)	1987	FURPS+ (Functionality, Usability, Reliability, Performance and Supportability(gives maintainability))	This hypothesis was not empirically validated.
Geoferry and Kemerer Approach	1991	Based on Cyclomatic Complexity Density	Sorry to say, these achievements have not been widely accepted.
Oman - Hagemeister Approach	1992	Halstead's Effort (ave E) , McCabe's Cyclomatic Complexity V(G), LOC(Lines of Code)Average CMT(Lines of Comment),Halstead's Volume(aveV)	Wasteful of resources, and requires unduly long times.
Li -Henry Approach based on coupling between classes	1993	DIT, LCOM ,NOC, WMC, CBO, RFC , DAC,NOM,	This method is practically very expensive
Coleman-Oman Approach	1994	Halstead's Volume (aveV) , McCabe's Cyclomatic Complexity V(G), LOC(Lines of Code)Average CMT(Lines of Comment)	This model is not validated for better acceptance.
Welker-Oman	1995	Based on Halstead's Volume(V), LOC(Total	Present the record of various

Approach(Improved Oman Approach)		Lines of Code) ,Cyclomatic Complexity V(g')	maintainability techniques or models
Dromey's Quality Approach	1995	Based on Modifiability, Testability as sub characteristics of maintainability	Not best suited to maintainability quantification at design phase
Muthanna Approach based on Polynomial Linear Regression	2000	Used Several Design Level Metrics	Which result has no any understandable relationship with object oriented design constructs and implementation.
Marcela Genero Approach	2003	Based on Structural Complexity metrics of UML class diagrams with understandability time, modifiability time, modifiability completeness, modifiability correctness and Size	Method was very difficult and has high Complexity.
Huffman Hayes Approach(Observe-Mine-Adopt (OMA))	2003	Used Maintainability product ,Perceived maintainability	Unfortunately maintainability measure only higher level of abstraction.
Lucca- Fasolino WAMM(Web Application Maintainability Approach)	2004	Apply Size metrics, Control coupling metrics ,data coupling metrics, Complexity metrics	Maintainability Prediction Model apply at later stage of SDLC
Kiewkanya Approach	2004	Based on Metrics Descriminant technique, Weighted Score Level Technique (understandability and Modifiability)	This model has a range of limitations from implementation point of view.
Hayes- Zaho Approach(MainPredMo)	2005	Used MI(Maintainability Index),LOC(Lines of Code), TCR(True Comment Ratio), AC(Attribute Complexity),CR(Comment Ratio)	This hypothesis was not empirically validated.
Aggarwal Approach(Artificial Neural Network ANN)	2005	Based on object oriented metrics(LCOM,DIT,WMC,NOC,RFC,DAC, MPC,NOM),Maintenance Effort	Regretful to say, these accomplishments have not been widely accepted.
S.C. Misra Approach	2005	Used Linear and multiple Regressions, Co-relation concept.	Wasteful of resources, and requires unduly long times.
WANG Li-jin, HU Xin-xin Approach	2005	Projection Pursuit Regression Approach(PPR)	This is practically very expensive
Koten-Gray Approach(Bayesian Network Maintainability Prediction Approach)	2006	Apply LOC(Line Of Code), SIZE1,LCOM(Lack of Cohesion of Methods), DAC(Data Abstraction Coupling), DIT(Depth of Inheritance),WMC(Weighted Method per Class),MPC(Message Passing Couple),NOM,SIZE2	This model is not validated for better acceptance.
Zhou -Leung Approach MARS (Multiple Adaptive Regression Splines)	2007	Compared with other Support vector Approaches, Multivariate linear regression Approaches, Regression tree Approaches and Artificial neural network Approaches and proved to be best.	Present the record of various maintainability techniques or models
Prasanth Ganesh Dalton Approach	2008	Based on FRT(Fuzzy Repertory Table) followed by Regression analysis	Not best suited to maintainability quantification at design phase
MO. Elish and KO Elish	2009	Treenet Approach using stochastic gradient boosting.	Which result has no any understandable relationship with object oriented design constructs and implementation.
Rizvi khan Approach(MEMOOD)	2010	Used LOC (Line of Code), MI(), NC (Number of Class),NA (Number of Attributes),NM (Number of Methods), Class Diagram	Method was very difficult and has high Complexity.
L Ping	2010	Based on Hidden Markov Approach	Unfortunately all above maintainability metrics measure

			only higher level of abstraction.
C Jin, JA Liu	2010	Based on Support vector machine	Maintainability Prediction Model apply at later stage of SDLC
Gautam Kang Approach (COMPOUND MEMOOD APPROACH)	2011	Used LOC (Line Of Code), DLOC (Difference Line of Code),MI (Maintainability Index), CC (Cyclomatic Complexity)	This model has a variety of limitations from implementation point of view.
Malhotra- Chug Approach	2012	Based on WMC, DIT, NOC, RFC,LCOM, MPC, DAC, NOM,SIZE2, SIZE1, CHANGE	This theory was not empirically validated.
Alisara Hincheeranan, Wanchai Rivepiboon	2012	Used hierarchies, abstraction, encapsulation, coupling, cohesion etc using Maintainability Quantification Tool (MET)	Unhappy to declare, these achievements have not been widely accepted.
Dubey et. al Approach	2012	Based on DIT, NOC, MPC, RFC, LCOM, DAC, WMC, NOM, SIZE1 and SIZE2 using Multi Layer Perceptron (MLP) neural network Approach	Wasteful of resources, and requires unduly long times.
Monika Saini, Mukti Chauhan	2013	Targeted various parameters including both source code metrics and design metrics	This approach is practically very expensive

V. IMPORTANT OBSERVATIONS

Following victorious completion of the literature analysis a numeral of key explanations can be enumerated as follows.

A. If quantify the software maintainability near the beginning that is design phase in the development life cycle may considerably improve the software quality and as well as customer happiness, and reduce overall cost, time and attempt of rework.
B. In arrange to dropping effort in quantifying maintainability of object oriented design study necessitate to identify a smallest set of maintainability factors for object oriented development process, which have positive impact on maintainability quantification.
C. Object oriented software properties are required to be documented and later than that the set of maintainability factors fitting at the design phase must be finalized.
D. Additional, maintainability metrics have to be selected at the design phase for the motivation that metric selection is a key step in maintainability quantification of object oriented design.

VI. CONCLUSIONS

After the above discussion our conclusion is that maintainability is a quality factor that attempts to quantify how much effort will be required for software maintaining and to quantify the complexity of causing a fault to result in a failure. The ambition of increasing the maintainability of software is not just to notice defects but more essentially, to notice defects as quickly as they are introduced. As a consequence, reducing the cost and time to fix the bug and producing higher quality maintainable software each one build of the release. Following a comprehensive evaluation procedure study found that reducing effort in quantifying maintainability of object oriented design is must in order to produce quality software in time and financial plan.

REFERENCES

[1] Pressman, R., "Software Engineering, A Practical Approach", 4th Edition, McGraw-Hill, 1997.
[2] M. Bruntink and A. Deursen, "Predicting Class Maintainability using Object-Oriented Metrics," *Proc. 4th IEEE International Workshop on Source Code Analysis and Manipulation SCAM'04*, 15 - 16 Sept., 2004,pp. 136 - 145, 2004.
[3] S.W.A. Rizvi and R.A. Khan, "A Critical Review on Software Maintainability Models," Proc. of the National Conference on Cutting Edge Computer and Electronics Technologies, 14 - 16 Feb. 2009, pp. 144 – 148,Pantnagar, India, 2009
[4] Abdullah, Dr, Reena Srivastava, and M. H. Khan. "Testability Estimation of Object Oriented Design: A Revisit.", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 8, Pages 3086-3090 August 2013.
[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] Abdullah, Dr, Reena Srivastava, and M. H. Khan. "Testability Measurement Framework: Design Phase Perspective.", International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 11, Pages 8573-8576 November 2014

[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, (Feb 1995) (vol. 21 no. 2), IEEE Transaction on Software Engineering, A Model for Software Product Quality

[16] Abdullah, Dr, Reena Srivastava, and M. H. Khan."Modifiability: A Key Factor To Testability", International Journal of Advanced Information Science and Technology, Vol.26, No.26, Pages 62-71 June 2014.

[17] S. Muthanna, K. Kontogiannis, K. Ponnambalaml and B. Stacey, "A Maintainability Model for Industrial Software Systems Using Design Level Metrics", In 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 maintainability. 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. In Proceeding of the 8th European Conference on Software Maintenance and Reengineering, pages 279– 287. IEEE Computer Society Press, 2004.

[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 EvolvingSystems", 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, Jan2006.

[24] K.K. Aggarwal, Y. Singh, P. Chandra and M. Puri, " Quantification 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 Quantification Model for Object-Oriented Software in 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

[36] Alisara Hincheeranan and Wanchai Rivepiboon," A Maintainability Quantification Model and Tool." International Journal of Computer and Communication Engineering, Vol. 1, No. 2, July 2012.

[37] 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.

[38] 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] Saini, Monika, and Mukti Chauhan. A Roadmap of Software System Maintainability Models "International Journal of Software and Web Sciences (IJSWS) www. iasir. net."

