Quantifying Maintainability of Object Oriented Design: An Organized Review

Mohit Kumar
Ph.D Scholar, Sai Nath University
Ranchi, Bihar, India
mohitsky25@gmail.com

Dr. Jarnail Singh
Professor, University Institute of Computing, Chandigarh University, Chandigarh

Dr. Abdullah
Assistant Professor, Department of Information Technology, Adigrat University Ethiopia-Africa

ABSTRACT
Maintainability is one of the most important quality indicators. Its correct estimation or evaluation, all the time make easy and improve the test and maintenance process. However, maintainability has always been an elusive concept and its correct quantification or evaluation is a difficult exercise. Researchers and practitioners have always argued that maintainability should be considered as a key attribute in order to assurance the software quality. It has been find out from systematic literature review that area researchers, quality controllers and industry personnel had made significant efforts to estimate software maintainability but at the source code level. Calculating maintainability at source code level directs to late arrival of desired information. An exact measure of software quality fully depends on maintainability quantification This paper shows the results of an organized literature review conducted to collect related evidence on maintainability quantification of object oriented software. In this paper, our objective is to find the known complete and comprehensive software maintainability quantification model and related framework for estimating the maintainability of object oriented software at an initial stage of development life cycle. Software maintainability has turn into one of the most significant concerns of the software industry. Maintainability is a key quality attribute of software systems.

Keywords:
Maintainability, Modularity, Reusability, Analyzability, Modifiability and Testability, Design phase, Object Oriented Design.

1. INTRODUCTION
Software maintenance is a microcosm of software development and has three- dimensional opinions. Maintenance may be defined by defining the four activities that are undertaken after a program is released for use. Software metrics provides an easy and inexpensive way to detect and correct the program to control the level of maintainability. One of the widely accepted approaches to control the software maintenance cost is the utilization of software metrics. Setting up measurement program and metrics standards will help in preventing failures before the maintenance process and reduces the requisite efforts during that phase. Internal metrics is highly correlated with the programmer's opinion of maintainability. Finally, according to ISO-9126 standard, maintainability consists of analyzability, changeability, stability and testability [1, 4, 6, 19]. Maintenance may be defined by defining the four activities [7, 10, 11, 16] that are undertaken after a program is released for use. First activity is the corrective maintenance that corrects uncovered errors after software is in use. Another activity is adaptive maintenance is applied when alterations in the outside background precipitate modifications to software. The third activity incorporates enhancements that are requested by customers and is defined as predictive maintenance. The fourth and last activity is the preventive maintenance, which improves future maintainability and provides a basis for future enhancements. To find out the deficiencies at in the early hours stage or early recognition of location where failure occurred is an important effort to mitigate the problem. Maintainability factor donate to 40-70% of the price of software products. Improved maintainability guide to reduced maintenance efforts and reduced price and time [10, 11, 36, 37, 39]. On the other hand, maintainability has always been an elusive concept and its correct quantification or evaluation a complex exercise. The majority of the studies measure maintainability or precisely the attributes that have impact on maintainability at the source code level. Our main passion is that it is for the duration of the analysis and design phase that maintainability analysis can yield the highest payoff: design decisions can be made to improve maintainability earlier than implementation starts. When the design meets the maintainability requirements, it can be implemented and the constraints added for maintainability enhancement of the design and are required to be verified before maintaining [2, 3, 5, 41, 43, 47, 48]. In maintainability have a number of regrettable consequences and as a result for many products and services is a severe warning. There is a general agreement among industry professionals and academicians to join together maintainability with the development life cycle in order to deliver protected, safe and reliable software inside time and budget [9, 14, 15, 17, 19, 20]. Our purpose is to present a comprehensive framework to help measuring and assessing maintainability in a practical manner, with a focus on the design stages of object-oriented development.

2. MAINTAINABILITY EVALUATION
Maintainability Quantification at the source code level is a good indicator of effort Quantification; it leads to the late appearance of information in the development process. A lot of Models exist, but no single model can take into custody a necessary amount of software characteristics. There are no particular model/tools that are applicable to all the circumstances. A choice to change the design in order to improve maintainability after coding has started may be very expensive and error-prone [21, 22, 24, 27, 28, 29, 31]. Despite the fact that estimating maintainability early in the development process may

significantly reduce the overall cost. This may shape a roadmap to industry personnel and study to assess, and preferably, quantify software maintainability in design phase. So reducing effort and improving software maintainability is a key objective in order to reduce the number defects that result from poorly designed software [30, 33, 34, 35,]. Therefore, it is an understandable fact that estimating maintainability early on the development procedure may significantly reduce maintenance cost, time, effort, and rework. The early Quantification of maintainability at design phase can yield the highest payoffs. On the other hand, the lack of maintainability at early stage may not be compensated during subsequent development life cycle.

3. MAINTAINABILITY AT DESIGN PHASE

Quantification Programming methodology is based on objects that involved functions and procedures, this concept allows individual object to organize and group themselves together into class. That requires the maintainability to be revealed because of the complex structure of object oriented development system because traditional testing approach is ineffective in this system. Practitioners incessantly support that maintainability should be planned early in the design phase. So it is important to identify object oriented design artifacts to quantify maintainability measures as early as possible in development life cycle. During identification of design factors which have positive impact on maintainability quantification, a pragmatic view should be considered. If we consider through all factors and procedures then they become more problematical, unproductive and time consuming. Therefore essential to categorize maintainability factors and measures which affect the movement positively and straight [8, 38, 40]. In order to estimating maintainability, its direct measures are to be identified. Design level aspects similar abstraction, encapsulation, inheritance, cohesion, coupling etc. will also be investigated keeping in view their impact on overall maintainability. This process identifies object oriented design constructs that are used during design phase of development lifecycle and serve to define a variety of maintainability factors. The contribution of each object oriented design characteristics is analyzed for improvement in design maintainability.

4. DESIGN PROPERTIES THAT INFLUENCES MAINTAINABILITY

Object oriented design properties overcome the negative aspect of procedure oriented design. Classes in object oriented design system provide an excellent structuring principle that allows a structure to be divided into well designed units which may then be implemented separately. Object oriented principles guide the designers what to support and what to avoid. Several measures have been defined in this approach so far to estimate object oriented design [42, 44, 46]. There are several essential qualities of object orientation that are known to be the basis of internal qualities of object oriented design and support in the context of maintainability quantification. These themes prominently include encapsulation, inheritance, coupling, and cohesion etc. One of the major advantage of having object orientation is its support for software reusability, which may be achieved either through the simple reuse of a class in a library or via inheritance among relationship [24, 26, 45].Object oriented design properties that have positive impact on maintainability quantification has been identified and consolidated chart for the same is given in Table 1.

Table 1: Object oriented design properties contributing in maintainability quantification: a critical look

Design Properties➔ Source/Study🡻	**Cohesion**	**Coupling**	**Encapsulation**	**Inheritance**	**Polymorphism**
Gregor et al. (1996)		✓	✓	✓	
Bruce & Shi (1998)	✓	✓		✓	✓
B. Pettichord (2002)	✓	✓		✓	
Baudry et al. (2002)		✓	✓		✓
M Bruntik (2004)	✓			✓	
S .Mouchawrab (2005)	✓	✓		✓	
E Mulo(2007)		✓	✓	✓	✓
Sujata et al. (2011)	✓			✓	
P. Malla et al. (2012)	✓	✓	✓	✓	
Nikfard et al. (2013)		✓	✓	✓	✓

5. CLOSELY RELATED WORK

5.1 Research Methodology

A systematic literature review is a technique of recognizing, estimating and understanding the existing research result significant to a particular research area or subject [22]. The study in research area has mainly divided into two categories primary and secondary studies. Primary study is an individual studies contributing to the research and secondary study is a systematic review of other research related to the research area, topic or observable fact of interest [22]. The enthusiasm for choosing systematic literature review as methodology of study are to sum up the existing body of knowledge regarding the research of concern, to recognize the gap in current research and to present framework/ contextual for additional examination. In this perspective, Study select the systematic review to sum up the existing concepts of maintainability factors and measurement in software engineering and apply that knowledge to build up a maintainability assessment framework/model for maintainability quantification

The justification for selecting this methodology is:

1. Data source selection
2. Search strategy development
3. Search string formation
4. Study selection criteria identification
5. Study quality assessment identification
6. Study extraction strategy identification

Opening from 1970s to 2020 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. 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. Study done by C Jin & JA Liu (2010) offerings the applications of support vector machine and unconfirmed learning in object oriented software maintainability estimation through metrics. In this study, the software maintainability predictor is performed 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. Subsequently, we found that support vector machine and clustering technique were supportive in emerging software maintainability predictor. Novel predictor can be used in the related software developed in the same background.

Work done by Gautama Kang (2011) emphasized dimension of the software maintainability close the beginning in the software development life cycle, mainly at the design period is very significant, and it support designers to integrate required improvement and corrections at design phase for improving software maintainability of the delivered software. Earlier MEMOOD model was developed which estimates the maintainability of the software system on the basis of object oriented metrics of software system. This work has suggested a multivariate linear model Compound "MEMOOD", which assessments the maintainability of class diagrams of software systems. Subsequently study make a comparison of MEMOOD model and Compound MEMOOD model through regression analysis and it is found that Compound MEMOOD Model gives better results with the given dataset. Moreover, no quantitative comparisons have been presented in this study. Study done by Alisara Hincheeranan *et.al* (2012) evaluated maintainability seeing maintainability and extensibility as two sub factors of maintainability. He stated measuring maintainability of software system at the design stage may facilitate a software designer must improves the maintainability of software before deliver to a customer. In this paper author developed the Maintainability Estimation Tool (MET) for a maintainability estimation of software system. This tool assist a software designer for improves the maintainability of class diagram in design phase and facilitate reduces the growing high cost of software maintenance phase. Moreover, no quantitative validation 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. Through 19 internal quality estimations, novelists empirically discover the effect of size, cohesion and coupling on class level maintainability. Acquired outcomes display that classes with enhanced qualities (greater cohesion values and lesser coupling and size values) have continuously improved 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 work done by R. & Chug A. (2014) offered a novel metric suite to overwhelmed the shortages and redefine the relationship amongst design metrics through software maintainability in data intensive applications. The proposed metric suite is estimated, analyzed using five proprietary software systems. The outcomes display that the suggested metric suite is very supportive for maintainability calculation of software systems in common and for data intensive software systems in specific. 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 Rajendra et. al. (2015) evaluated and authenticated the model for software maintainability based on quality factors flexibility and extendibility [39]. The outcomes they arrived stood important but by other factors newer models for maintainability with better-quality outcomes could be proposed. Study done by Ruchika Malhotra et.al. (2016), in their research paper assembled a methodical analysis of studies on software maintainability amongst the years 1991 to 2015[31]. The authors organized and scrutinized the effort on maintainability by tangents of design metrics, tools, algorithms, data sources and so on. They concise that design metrics was

still the greatest preferred choice to capture the features of any given software before installing it additional in prediction model for formative the corresponding software maintainability.
Celia Chen et al. (2017) in their work stressed the vast level of cost saving in software by understanding the significance of maintainability, and recommended replies to queries of decision concerning what portions of software to be reused, what portions to be redeveloped, the theoretical valuation of effort requisite to do so and thus giving pointers as how to decrease overall budgets [32].

5.2 Analysis and Comparison

Table 2: An Organized Assessment of Maintainability Models Consider by Various Researcher

Study done by	**Year**	**Maintainability quantification method**	**Development Stage**	**Authentication**
Dromey's Quality Model	1995	Quality Model	Code Level	Theoretical justification
Muthanna et al.	2000	Model based on Polynomial Linear Regression	Design Phase	No Validation
Huffman Hayes et al.	2003	Observe Mine Adopt (OMA) Based on Maintainability product	Code Level	Yes
Lucca-Fasolino WAMM	2004	Web Application Maintainability Model	Web based Approach	Web based Approach
Hayes Zaho	2005	(Main Pred Model) LOC (Lines of Code), TCR (True Comment Ratio)	Code level	No Validation
Koten-Gray	2006	Bayesian Network Maintainability Prediction Model	Code level	Yes
Zhou - Leung MARS	2007	Multiple Adaptive Regression Splines	Design Phase	No Implementation.
Prasanth Ganesh and Dalton	2008	With the help of FRT(Fuzzy Repertory Table)	Design Phase	Not Validated
MO. Elish and KO Elish	2009	Produced Tree net model using stochastic gradient boosting	Code level	Not Validated
C Jin & JA Liu	2010	Based on Support vector machine	Code level	Based on vector machine
S. Rizvi et al.	2010	MEMOOD Model	Design Phase	No Validation
Gautama Kang	2011	Compound Memood Model	Design Phase	Not Validated
Alisara et al.	2012	Maintainability Estimation Tool (MET)	Code level	No Validation
Al Dallal, J.	2013	Object oriented class maintainability calculation via internal quality attribute.	Design and code level	Not Validated
R. & Chug A.	2014	A Metric Suite for Predicting Software Maintainability in Data Intensive Applications.	Design Phase	Based on Metrics
Rajendra et. al.	2015	Maintainability based on quality sub factors	Design Phase	Based on regression line
Ruchika Malhotra et.al.	2016	Maintainability by tangents of design metrics	Not clear	Not Validated
Celia Chen et al.	2017	Importance of software maintainability	SDLC	Theoretical estimation
Hadeel Alsolai et al.	2019	Maintainability in Object Oriented Systems Using Ensemble Methods	SDLC	Validated

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. If we can calculate the maintainability at the near the beginning stages of the software development, the price of the software can be reduced. A range of software quality models are considered. After studying these models a comparison table is made which give you an idea about the various models and their uniqueness (Table 2). All models have some characteristics like Portability, Usability, Modifiability, Maintainability, etc. as marked in the Table 2.1. However here

ISSN: 2350-0557, Volume-6, Issue-6, November 2019
www.ijirem.org

the main importance is given on maintainability characteristics of software quality models for the reason that it is the factor which have an effect on the software system the most.

A variety of software quality models, like Dromey's, McCall's, FURPS+ ISO/IEC- 2510 etc., are existing in which maintainability is defined. Maintainability is evaluated by various studies through several software quality sub characteristics such as Testability, changeability, stability and Maintainability.

6. SOFTWARE QUALITY MODELS

Table 3: A comparison of four Quality Models and ISO/IEC- 2510

Approach➔ / **Factors🡻**	**McCall**	**Boehm**	**Dromey**	**FURPS**	**ISO/IEC- 2510**
Correctness	✓		✓		
Integrity	✓		✓		✓
Reusability	✓	✓			✓
Understandability			✓		
Modularity	✓				✓
Effectiveness	✓	✓	✓		✓
Analyzability	✓				✓
Maintainability	✓	✓	✓	✓	✓
Adaptability				✓	✓
Modifiability		✓	✓		✓
Compatibility				✓	✓
Testability	✓				✓
Installability			✓	✓	

After the above discussion our conclusion is that maintainability is a quality factor that attempts to predict how much effort will be required for software maintenance. The goal of increasing the maintainability of software is not just to detect defects but more importantly, to detect defects as soon as they are introduced. Consequently, reducing the cost and time to fix the bug and producing higher quality maintainable software each build of the release. In order to obtain consistent and correct quantification 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.

7. RELEVANT FINDINGS

After successful completion of the literature review a number of important explanations can be enumerated as follows.

i. If we estimate the software maintainability at an early stage that is design phase in the software development process may significantly improve the software quality and as well as client happiness, and decrease overall cost, time and effort of rework.

ii. In order to reducing effort in measuring maintainability of object oriented design we require to recognize a minimal set of maintainability factors for object oriented development procedure, which have optimistic impact on maintainability quantification Object oriented software characteristics are required to be recognized and after that the set of maintainability metrics appropriate at the design phase should be finalized.

iii. Further, maintainability metrics have to be chosen at the design phase for the reason that metric selection is an important step in maintainability Quantification of objects oriented design.

8. CONTRIBUTION

The most important contribution of this paper is in the field of maintainability quantification. We have accompanied an organized review in this field. The dissimilar factors of maintainability and quantification for these factors are identified. Overall contribution is listed as follows:

i. Systematic Literature Review
ii. A complete step by step improvement of the systematic review procedure is described. It will help to further study as a reference for undertaking SLR.
iii. Recognition of key papers related to the maintainability study in software engineering domain
iv. Discovery of maintainability factors and quantification in the recent domain of OOD
v. Identification and arrangement of different concepts about the software maintainability in the present software engineering domain.
vi. To propose a software maintainability framework to assist the self-assessment for designers to identify software maintainability factors.
vii. Structure and well defined assessment process for finding factors from high level to lower measurable level.

9. CONCLUSION

With growing complexity, pervasiveness and criticality of software, building reliable and quality end software is becoming more and more challenging. Moreover, the advancement in the software development process has been accelerated drastically in the last couple of decades. As a result, the complexity of applications and environments has been substantially increased and schedules have been pinched. Under these environments, software quality inclines to agonize. In the face of intense competitive pressure, a comprehensive and rational strategy to achieve high maintainability will be a strategic advantage-not a bottleneck. The foregoing analysis implies that maintainability results from good Software Engineering practice and an effective software process. A number of approaches have been proposed in the literature for measuring software maintainability. An investigation of the considerable literature shows that greatest efforts have been place at the later stage of software development life cycle. A resolution to modify the design in order to improve maintainability after coding has started is very costly and error-prone. After the above conversation our conclusion is that maintainability is a quality factor that attempts to calculate how much effort will be required for software maintaining and to estimate the trouble of causing a fault to outcome in a failure. The goal of increasing the maintainability of software is not just to detect defects but more importantly, to detect defects as soon as they are introduced. Thus, reducing the cost and time to fix the bug and producing higher quality maintainable software each build of the release. After an exhaustive review process we found that reducing effort in measuring maintainability of object oriented design is must in order to deliver quality software within time and budget.

REFERENCES

1. K.K. Aggarwal, Yogesh Singh. *New Age International, Jan 1,* 2005 - Software engineering.
2. Singh, Hardeep, 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. Singh, Pradeep Kumar, and Om Prakash Sangwan. "Aspect Oriented Software Metrics Based Maintainability Assessment: Framework and Model." (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. (1985), Automated Software Quality Assurance. IEEE Trans. Software Eng., 11Bi, 9: 909-916.
9. Grady, Robert, Caswell, Deborah (1987), *Software Metrics: Establishing a Company-wide Program*. Prentic 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. 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", 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 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, Jan2006.
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. Sub has 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 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. Celia Chen , Alfayez R ,Kamonphop Srisopha and Lin Shi, Why Is It Important to Measure Maintainability and What Are the Best Ways to Do It?, IEEE/ACM 39th
33. International Conference on Software Engineering Companion (ICSE-C), July 2017.
34. 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.
35. 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.
36. Malhotra *et al.* "Software Maintainability Prediction using Machine Learning Algorithms." Software Engineering: An International Journal (SEIJ), Vol. 2, No. 2, SEPTEMBER 2012
37. Alisara Hincheeranan and Wanchai Rivepiboon," A Maintainability Estimation Model and Tool." International Journal of Computer and Communication Engineering, Vol. 1, No. 2, July 2012.
38. 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.
39. Laxmi Shanker Maurya *et.al,"* Maintainability assessment of web based application.'', *Journal of Global Research in Computer Science,* Vol 3, No. 7, July 2012.
40. Rajendra Kumar and Dhanda N, Maintainability Measurement Model for Object-Oriented Design, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 5, May 2015.
41. 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.
42. Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M., (1978) *Characteristics of Software Quality,* North Holland.
43. ISO 9126-1 Software Engineering - Product Quality - Part 1: Quality Model, 2001.
44. Grady, Robert, Caswell, Deborah (1987), *Software Metrics: Establishing a Company- wide Program*. Prentice Hall. pp. p. 159. ISBN 0138218447.
45. Sneed, H., Mercy, A. (1985), Automated Software Quality Assurance. IEEE Trans. Software Eng., 11Bi, 9: 909-916.
46. Sommerville, I. (1992). Software Engineering. 4th ed. New York, Addison- Wesley.
47. Hordijk, Wiebe, and Roel Wieringa. "Surveying the factors that influence maintainability: research design." *ACM SIGSOFT Software Engineering Notes*. Vol. 30. No. 5. ACM, 2005.
48. Larrucea X., Santamaria I., O'Connor R., Messnarz R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2018. Communications in Computer and Information Science, Vol 896, pp. 492-503. Springer, Cham.

