ISSN: 2321-7782 (Online)
Volume 2, Issue 10, October 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Object Oriented Design for Testability: A Systematic Review

Shweta Srivastava[1]
M.Tech. Scholar, Department of Computer Science, R.K.G.IT, Ghaziabad, India.

Dr. Majhar Khaliq[2]
Assistant Professor, Department of Computer Science. K.M.C.A.F.U., Lucknow, India

Abstract: ***Testability is one of the most important quality indicators. Its correct measurement or assessment, all the time make easy and improve the testing procedure. On the other hand, testability has always been an intangible concept and its correct measurement or evaluation is a complicated exercise. Researchers, quality controllers and practitioners have always argued that testability should be considered as a key attribute in order to assurance the quality software. A perfect measure of software quality totally depends on testability measurement. This paper presents the outcomes of a systematic literature review conducted to collect facts on software testability of object oriented design. Improving software testability is key objective in order to reduce the number of defects that result from poorly designed software.***

Keywords: Software Testing, Testability, Quality, Object Oriented Characteristics, Quality Factors, Testability Factors.

I. INTRODUCTION

Software development processes normally focus on keep away from errors, identifying and correcting software faults that do occur, and predicting software reliability after development. It is well understand that delivering quality software is no longer an advantage but a necessary factor. Unhappily, the majority of the industries not only fails to deliver a quality product to their customers, but in addition do not recognize the important quality attributes [1]. Software testing is an important branch of both the software development lifecycle and quality assurance activities. Software testing is one of the most famous ways of promising excellence of software system; the usefulness of testing decides the quality of released software. On the other hand, testing has now turned into a boring job and a pricey activity, for the reason that the size and complexity of software is growing rapidly. Software testing is n financial problem directly intertwined with nearly all major technical issues in Software Engineering.

Testability is a quality factor; its measurement or assessment can be used to expect the amount of effort required for testing and facilitate allocating required resources. There is no understandable definition to 'what aspects of software are actually related to testability' [2]. Current software practice places a strong importance on unit testing, to the degree that the amount of test code produced on a project can exceed the amount of actual application code required. This demonstrates the importance of testability as a characteristic of software [3].

II. TESTABILITY ESTIMATION OF OBJECT ORIENTED SOFTWARE

However, testability has always been an elusive concept and its correct measurement or evaluation a difficult exercise. The majority of the studies assess testability or specifically the attributes that have impact on testability at the source code level. It has been inferred from the literature on testability factors that there is a serious need of proposing a generally accepted set of the factors affecting software testability [4, 22].

TABLE I

Testability Factors / Author/Study ↓	Extendability	Effectiveness	Flexibility	Functionality	Understandability	Modifiability
[1]Binder (1994)	√	√	√	√		√
[2]Bach (1999)		√			√	√
[3]Jungmayr(2002)	√		√	√		√
[4]Wang(2003)	√	√	√	√	√	
[5]Jungmayr (2003)	√			√		
[6]Jerry(2005)	√	√		√	√	√
[7]E Mulo(2007)	√	√		√		
[8]Dino Esposito(2008)		√		√		√
[9]Nazir &Khan(2009)	√	√	√	√		√
[10]Nazir et al. (2010	√	√	√	√		√
[11]P.Malla &G(2012)				√	√	√
[12]Nazir et al. (2012)	√	√	√	√		√
[13]P.Nikfard(2013)		√		√		√

Testability Factor Consider by various Experts: A Close Look

III. IMPROVING TESTABILITY AT DESIGN PHASE

Testability assessment at the source code level is a good indicator of effort estimation; it leads to the late appearance of information in the software development process. A choice to change the design in order to improve software testability after coding has started may be very costly and error-prone. Despite the fact that estimating testability near the beginning in the development process significantly reduces the overall cost and rework. Early estimation of testability mainly at design phase form a roadmap to industry persons and researchers to review, and preferably, measure and improve software testability at design phase. So reducing effort and improving software testability is a key objective in order to reduce the number of defects that result from defectively designed software.

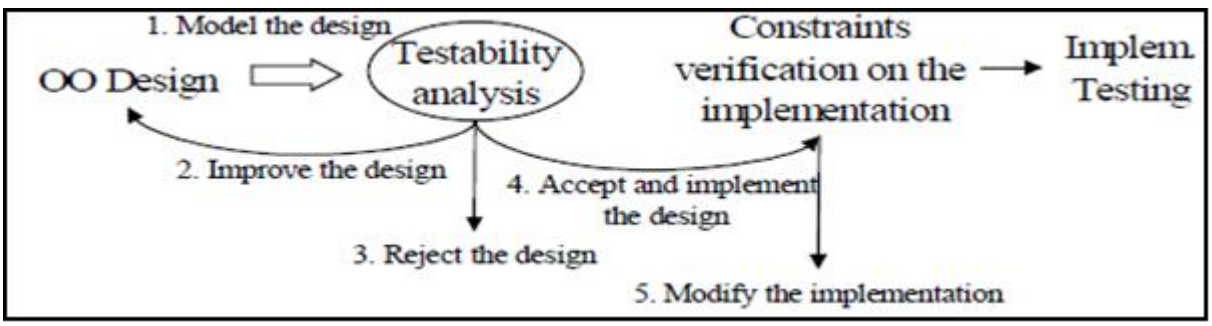

Figure 1. [15]

IV. OOD PROPERTIES

Object orientation is now in extensive adopt by the software industry, regardless of the truth that technology is not grown-up enough from the testing point of view .Regardless of the fact that object oriented tools has now been broadly accepted by the

software industry, only a small number of research studies have been dedicated to explore the concepts of testability in object oriented systems. Numerous developments on the measurement of testability, design for software testability have been listed in the literature [9, 10, 16, 17 and 18].

TABLE III

Design Parameters / Author/Study ↓	Cohesion	Coupling	Encapsulation	Inheritance	Abstraction
MC Gregor et al. (1996)			√	√	
Bruce & Shi(1998)		√		√	
B.Pettichord(2002)		√			
Baidry et al.(2002)		√			
M Bruntik (2004)				√	
S.Mouchawrab (2005)	√	√		√	
I.Ahson et al.(2007)	√	√	√	√	
Nazir et al.(2010)	√	√	√	√	
Suhel et al.((2012)	√	√	√	√	√
Khan et al. (2012)	√	√	√	√	
Nikfard &Babak(2013)		√	√	√	

OO Design Constructs Contributing in Testability Improvement at Design Phase: A Critical Look

V. RELATED WORKS

Broad range of testability estimation models have been planned in the literature within last two decades. A number of testability models/methodologies were proposed to facilitate the designers for measuring and improving the testability of object oriented software to produce superior and improved software systems. Opening from 1975 to 2014 a variety of testability assessment models or techniques was developed.

Testability is an elusive concept. It is very complicated to find a clear view on all the prospective factors that have an effect on software testability for supporting the test process. The study on software testability primarily comes into view in 1975. It is accepted in McCall and Boehm software quality model, which build the foundation of ISO 9126 software quality model. Since 1990s, software engineering society began to initiate quantitative research on software testability. Software testability evaluation has been a vital research direction since 1990s and became extra pervasive in 21st century [41].A number of researchers addressed software testability, but in the background of conventional structured design. The question of testability

has been revitalized with the object orientation [22, 23, 24]. Despite the fact that object oriented technology has now been broadly accepted by the software industry, only a small number of research studies have been dedicated to travel around the concepts of testability in object oriented systems.

A lot of developments on the assessment of software testability and design for testability have been reported in the available literature [12, 16, 20, 25 and 26]. Sorry to say, these achievements have not been extensively accepted and hence, not been adopted into practice by software industry [1, 27]. Following sections analytically summarize some of the appropriate important efforts made by researchers in this area.

In 1993, Voas and Miller [28] draw concentration to software testability metric that are fully based upon inputs and outputs artifacts of a software module. To compute testability, they wished-for PIE (propagation, infection as well as execution) study procedure [22], on the other hand measuring testability all the way through the PIE analysis method was very multifarious and has high complexity. Due to these restrictions this technique is not adopted by industry personals and researchers to extend their works.

In 1994, Binder had accomplished a vast work highlighting the significance of improving software testability in system development life cycle. He projected a fishbone model on behalf of the key factors of testability. Fishbone model usually include, software testability is an outcome of six factors namely:(1) Characteristics of the representation (2) Characteristics of the implementation (3) Built-in test capabilities (4) The test suite (test cases and associated information) (5) The test support environment (6) The software process in which testing is conducted [24]. But unluckily all above testability factors evaluate only higher level of abstraction. Which result has no any understandable relationship with object oriented design constructs and implementation.

In 1998, Bruce and Haifeng Shi [29] decorated the object oriented design testability factors that have an effect on testability, and showing their impact for improving software testability of object oriented design. They intended a model for testability measurement with the help of single testability factor and design level constructs. But this skeleton has various limitations from implementation point of view.

In 2002, Jungmayr [30] presented a novel thought for estimating software testability through integration testing and emphasized only these components. He recognized local dependencies that optimistically contributing and answerable for overall testability. Jungmayr concept *used* reduction metric to calculate the effect of individual factors of software testability to find out required testability metric.

In 2004, *Bruntink and van Deursen* [14] presents a collection of metrics for exploring the testability of object oriented Java system, and acknowledged testability factors throughout source code metrics. In order to get better testability, a choice to change the design after coding was started it very costly and error prone.

In 2005, Baudry et al. highlighted the significance of individual types of class combination on testability metrics. And set up a relation of coupling and class interaction metrics that confirm testability [31]. On the other hand, this hypothesis was not empirically validated and adopted by industry.

In 2005, for assessing software testability, Jerry and Ming presented a model that is based on pentagon shaped and analytical approach [26].

In 2007, Mulo incorporate the significance of testability measurement all the way through the software development life cycle [32]. Estimating testability during the entire development life cycle is very costly and error prone and required added effort.

In 2009, Jianping Fu & Minyan Lu planned a request oriented scheme for software testability measurement [33].This technique selects correct elements from a self contained software testability measurement framework according to the different

measurement requirements to complete all kinds of software testability measurement. Unluckily, these achievements have not been widely accepted and hence, have not been adopted in practice by the industry.

In 2011, Fadel Toure [34] offered a new approach for improving testability of the software via software reliability growth models. On the other hand, the different approaches discussed for improving and measuring software testability were hypothetical and the quantitative measure of enhancement of testability was not specified.

In 2012, Badri, Mourad, and Fadel Toure, focused on establishing the correlation among object oriented metrics and testability of classes in terms of required testing effort [35]. For this they performed an empirical investigation using data collected from three Java software systems for which JUnit test cases exist. To take into supervision testability of classes, they used different metrics to compute the corresponding JUnit test cases. The metrics associated to the JUnit test cases were used, in fact, to systematize the classes in two categories in terms of required testing effort: high and low. In this work, testability has been investigated from the perspective of unit testing.

In 2013, Pizzi, Nick J. [36] intended fuzzy classification approach. In this approach a large collection of classifiers is presented with subsets of the software metric features. Subsets are preferred stochastically using a fuzzy logic based sampling method. The classifiers after that calculate the quality, exclusively the class label, of every software object. Fuzzy combination is applied to the results from the most truthful individual classifiers. These classifiers estimate the quality mostly at the class level, which direct to no understandable relationship with the software metric features that are based on design artifacts and the implementation level.

In 2013, Tiwari, Rajeev and Noopur Goel [37] projected reuse-oriented test approaches, which are used to decrease the testing effort. Supplementary, he stated the state-of-the-art in reuse oriented test approaches employed in reuse oriented software development processes. But this approach is not broadly suitable for new products. The reuse-oriented approach is not forever useful in its pure form, for the reason that a full collection of reusable components may not be available.

Kaur, Kiranjit, and Sami Anand [40] produced a multivariate linear model, this model quantify the maintainability of a class diagram. He primarily is paying consideration only on maintainability estimation in terms of testability, reliability, portability. These metrics support a software designer for improving the maintainability of a class diagram at the design phase of development life cycle.

After a reconsideration tour we found that a variety of methods or techniques are available in the literature for improving and estimating software testability. A review of the testability estimation of object oriented design shows that greatest effort center of attention at the later stage of software development life cycle. On the other hand, current developments in software design management repeatedly advocate integrating software testability at design phase. This in turn will help the designers to improve quality and security of software and significantly reduce the overall development cost and rework of development life cycle for producing high quality maintainable, testable and reliable software.

VI. CRITICAL OBSERVATIONS

Above successful accomplishment of the systematic literature review, a number of key explanations can be enumerated as follows.

- An early estimation of software testability mainly at design phase in the software development process significantly improve the software quality and decrease overall development cost, time and effort of rework.
- In order to improve testability of object oriented design and reduce effort in measuring testability of object oriented design we have need of to recognize a minimum set of testability factors which have positive impact on testability measurement at design phase.

- Object oriented software characteristics are mandatory to be recognized and after that the set of testability factors suitable at the design phase should be finalized.
- Further, testability metrics have to be preferred at the design phase, for the motivation that metric selection is an essential step in testability estimation of objects oriented design.

VII. CONCLUSION

Quite a lot of approaches have been proposed in the literature for improving and measuring software testability. A study of the appropriate literature is evidence for that greatest hard work on testability measurement has been put at the later phase of software development life cycle. A promise to modify the design in order to recover testability after coding has started is extremely costly and error-prone. For this explanation, it is a noticeable fact that estimating testability near the beginning in the development system significantly decrease testing time, effort, rework and cost. The timely estimation of software testability at design phase can yield the uppermost payoffs. On the other hand, the lack of testability at an early stage may not be remunerated throughout the successive development life cycle.

After the above conversation our conclusion is that testability is a quality factor that attempts to compute that how much effort will be required for software testing. Later than an exhaustive analysis process study found that improving software testability and reducing effort in measuring testability of object oriented design is necessity in order to deliver high quality software within time and budget.

VIII. CONTRIBUTION

The most important contribution of this study is in the field of object oriented design testability improvement and measurement. Study has conducted a systematic review in this field. The unrelated testability factors and measurement are recognized. Overall contribution is listed as follows:

- Conduct systematic literature review to collect facts based on testability improvement and estimation of object oriented software.
- A complete step by step progress of the systematic review procedure is described. It will help to further researchers as a reference for undertaking SLR.
- Identification of key papers related to the testability improvement and estimation.
- Finding of testability factors and measurement in the current domain of OOD.
- Categorization and kind of different concepts about the software testability in the present software engineering field.

References

1. R A Khan, K Mustafa, S I Ahson, “Software Quality Concepts and Practices”, Narosa Publishing House Pvt. Ltd., 2006.
2. Mazhar Khaliq, Riyaz A. Khan, M. H. Khan, “Software quality Measurement: A Revisit”, Oriental Journal of Computer Science & Technology, vol.3 (1), 05-11, June 2010.
3. R A Khan, K Mustafa, "MQMOOD: Metric Based Model for object oriented Design Quality Assessment", Information Technology Journal 4, 2005.
4. M. Sarker, “An overview of Object Oriented Design Metrics”, Master Thesis, Department of Computer Science, Umeå University, Sweden, 2005.
5. Abdullah, Dr, Reena Srivastava, and M. H. Khan. "Testability Estimation of Object Oriented Design: A Revisit."
6. Abdullah, Dr, Reena Srivastava, and M. H. Khan. "Modifiability: A key factor to testability
7. R A Khan, K Mustafa, "MQMOOD: Metric Based Model for object oriented Design Quality Assessment", Information Technology Journal 4, 2005.
8. M. Sarker, “An overview of Object Oriented Design Metrics”, Master Thesis, Department of Computer Science, Umeå University, Sweden, 2005.
9. B. Grady, “Object-Oriented Analysis and Design with Applications”, 2nd edition, Benjamin Cummings, 1994.
10. Testability Estimation Model (TEMOOD): M. Nazir & R.A.Khan, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, Vol. 85, Meghanathan, Natarajan; Chaki, Nabendu; Nagamalai, Dhinaharan (Eds.), Volume 85, Part 3, LNICST, Springer-Verlag, 2012, pp 178-187, (ISBN 978-3-642-27307-0). January 2012.

11. Cinnéide, M.O. Boyle, D.; Moghadam, I.H.," Automated Refactoring for Testability ", Software Testing, Verification and Validation Workshops (ICSTW), Page(s): 437 – 443, IEEE Fourth International Conference Ireland, March 2011.
12. Nazir M., Khan Raees. A.," An Empirical Validation of Understandability Quantification Model", Journal Procedia Technology, 2nd International Conference on Computer, Communication, Control and Information Technology, Volume 4, Pages 772–777, 2012.
13. Manu Phogat, Dharmender Kumar, "Testability of Software System "IJCEM International Journal of Computational Engineering & Management, Vol. 14, ISSN (Online): 2230-7893, October 2011.
14. M. Bruntink and A. V. Deursen, "Predicting Class Testability using Object-Oriented Metrics," Proc. IEEE International Workshp on Source Code Analysis and Manipulation, pp. 136-145, 2004.
15. S. Mouchawrab, L. C. Briand, and Y. Labiche, "A measurement framework for object-oriented software testability", Volume 47, Issue 15, Pages 979-997.December 2005.
16. Improving the Testability of Object-oriented Software during Testing and Debugging Processes, Sujata Khatri, R.S. Chhillar, V.B.Singh, International Journal of Computer Applications (0975 – 8887) Volume 35– No.11, December 2011.
17. S. Jungmayr, "Design for Testability", CONQUEST 2002, pp. 57-64.
18. S. Mouchawrab et al, "A measurement framework for object-oriented software testability," Carleton University, Technical Report, SCE-05-05, year 2005.
19. Jianping, Fu, Liu Bin, and Lu Minyan."Present and future of software testability analysis." Computer Application and System Modeling (ICCASM), 2010 International Conference on. Vol. 15. IEEE, 2010.
20. Nazir, Mohd, and Raees A. Khan. "Testability Estimation Model (TEMOOD)." Advances in Computer Science and Information Technology. Computer Science and Information Technology. Springer Berlin Heidelberg. 178-187, 2012.
21. Abdullah, Dr, Reena Srivastava, and M. H. Khan. "Testability Estimation of Object Oriented Design: A Revisit."
22. Voas and Miller, "Software Testability: The New Verification". IEEE Software. Vol. 12(3), p. 17-28, 1995.
23. J.M. Voas. "Object-Oriented Software Testability". In proceedings of International Conference on Achieving Quality in Software, January 1996
24. R.V. Binder, "Design for testability in object-oriented systems". Communications of the ACM. Vol. 37(9), p. 87-101, 1994.
25. M. Bruntink and A. V. Deursen, Predicting class testability using object oriented metrics, in Proc. IEEE international Workshop on Source Code Analysis and Manipulation, pp. 136-145, 2004.
26. J. Gao and Ming-Chih Shih, component testability model for verification and measurement, In Proc. of the 29th Annual International Computer Software and Applications Conference, pages 211–218. IEEE Comp Society 2005.
27. M. Nazir, Khan R A & Mustafa K. (2010): Testability Estimation Framework, International Journal of Computer Application, Vol. 2, No. 5, pp.9-14. June 2010
28. Voas and Miller, Semantic metrics for software testability, Journal of Systems and Software, Vol. 20 (3), pp. 207-216, 1993.
29. Bruce W.N.Lo and Haifeng Shi, A preliminary testability model for object-oriented software, in Proc. International Conf. on Software Engineering, Education, Practice, Pages 330{337. IEEE. 1998.
30. Jungmayr, S. Testability Measurement and Software Dependencies. In Proceedings of the 12th International Workshop on Software Measurement, pp. 179–202, October 2002.
31. Baudry and Traon, Measuring Design Testability of a UML Class Diagram. Information and Software Technology, 47(13):859–879, 2005.
32. E. Mulo, "Design for Testability in Software Systems", Master's Thesis, 2007. URL:swerl.tudelft.nl/twiki/pub/Main/ResearchAssignment/RA-Emmanuel-Mulo.pdf
33. Fu, Jianping, and Minyan Lu. "Request-Oriented Method of Software Testability Measurement." Information Technology and Computer Science, ITCS 2009. International Conference on. Vol. 2. IEEE, 2009.
34. Improving the Testability of Object-oriented Software during Testing and Debugging Processes, Sujata Khatri, R.S. Chhillar, V.B.Singh, International Journal of Computer Applications (0975 – 8887) Volume 35– No.11, December 2011.
35. Badri, Mourad, and Fadel Toure. "Empirical Analysis of Object- Oriented Design Metrics for Predicting Unit Testing Effort of Classes." Journal of Software Engineering and Applications 5.7: 513-526, 2012.
36. Pizzi, Nick J. "A Fuzzy Classifier Approach to Estimating Software Quality." Information Sciences (2013). Volume 241, Pages 1–11, 20 August 2013.
37. Tiwari, Rajeev, and Noopur Goel. "Reuse: reducing test effort." ACM SIGSOFT Software Engineering Notes 38, no. 2: 1-11, 2013.
38. Panigrahi, Chhabi Rani, and Rajib Mall. "An approach to prioritize the regression test cases of object-oriented programs." CSI Transactions on ICT: 1-15, 2013.
39. Amid, Amin, and Somaye Moradi. "A Hybrid Evaluation Framework of CMM and COBIT for Improving the Software Development Quality." 2013.
40. Kaur, Kiranjit, and Sami Anand. "A Maintainability Estimation Model and Metrics for Object-Oriented Design (MOOD)." International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 2.5: pp-1841, 2013.
41. L. Zhao, "A new approach for software testability analysis", International Conference on Software Engineering, Proceeding of the 28th international conference on Software Engineering, Shanghai, pp. 985–988, 2006.

AUTHOR(S) PROFILE

Shweta Srivastava, working as Sr. Lecturer in Babu Banarasi Das University, Lucknow, U.P. She has completed MCA from IGNOU 2003, and at present doing M.Tech. in Computer Science from UPTU Lucknow. She has approx 8 years of rich teaching experience at UG and PG level. Her research interest includes software quality, and Software testability.

Dr. Majhar Khaliq, Assistant Professor, Department of Computer Science at K.M.C.A.F.U., Lucknow, India. Obtained his MCA degree from Aligarh Muslim University (Central University).Later he did his Ph.D. in Software Engineering from Integral University, Lucknow. He has around 11 years rich teaching experience at UG and PG level. His area of research is Software Engineering. Dr. Majhar published numerous articles, several papers in the National and International Journals and conference proceedings.