ISSN: 2321-7782 (Online)
Volume 2, Issue 12, December 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Testability Estimation of Object Oriented Software: A Systematic Review

Rishabh Srivastava[1]
Research Scholar
Department of Computer Science
Goel Institute of Technology & Management
Lucknow (U.P.) – India

Namrata Dhanda[2]
Associate Professor and head of CS/IT
Department of Computer Science
Goel Institute of Technology & Management
Lucknow (U.P.) – India

Siddharth Lavania[3]
Assistant Professor
Department of Computer Science
Goel Institute of Technology & Management
Lucknow (U.P.) – India

Abstract: ***Testability is an elusive idea, its precise estimation or assessment is a hard exercise. It is very tough to obtain an understandable view on all the probable factors that have positive impact on software testability. Researchers, practitioners and quality controllers have always argued that testability should be measured as a key attribute in order to promise the quality software. A faultless measure of software quality completely depends on testability measurement. The support provided by software testability is important throughout development life cycle and quality assurance. This paper presents the outcomes of a systematic literature review conducted to collect facts on software testability of object oriented design. Finally study does a comparative analysis on software testability proposed by various experts/researchers including their contribution and limitation. Improving software testability is key objective in order to reduce the number of defects that result from poorly designed software.***

Keywords: Software Testability; Testability Quantification; Object Oriented Design Characteristics; Software Quality; Software testing.

I. INTRODUCTION

Software development processes usually focus on keep away from errors, identifying and correcting software faults that do happen, and predicting software reliability after development. It is well know that delivering superiority software is no longer an advantage but an essential factor. Unhappily to say, the bulk of the industries not only fails to carry a quality product to their customers, but in addition do not identify the important quality attributes [1]. Software testing is a main branch of both the software development lifecycle and quality promise activities. Software testing is only of the most well-known ways of promising fineness of software system; the effectiveness of testing decides the quality of released software. On the other hand, testing has now turned into an uninteresting job and an expensive activity, for the cause that the size and complication of software is rising speedily. Software testing is an economic problem straightforwardly intertwined with almost all main technical issues in Software Engineering.

Testability is a quality factor; its measurement can be used to guess the quantity of effort required for testing and smooth the progress of allocating required resources. There is no logical definition to 'what aspects of software are in reality related to testability' [2]. The decisive goal of software engineering is to make quality oriented software that is trustworthy, testable, devoted, and produced in time, monetary plan and as well fulfilled its specific necessities. Most vital intention will be making the testing process easy and detecting the defects in victorious and positive way. With the increasing value of testability in

software, it is easier to arise the incorrect output in case of existence of defect in the software [18], [19].The testability approach increases the opportunity of revealing the faults, finally making software fault detection process easier.

II. Software Testability

Software testability is defined by IEEE as "the degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been meet" [28]. Testability is an unsolved concept. It is very hard to obtain a logical view on all the probable factors that have an effect on software testability.ISO has defined software testability as a functionality and it defines functionality as "the set of attributes of software that bear on the effort needed to validate the software product" [31]. A choice to renovate the design in order to improve software testability after coding has started may be extremely costly and error-prone. Regardless of the fact that estimating testability close to the beginning in the development process considerably reduces the overall cost and rework. The study on software testability initially appeared in 1975. It is accepted in McCall and Boehm software quality model, which formulate the basis of ISO 9126 quality model. From the time when 1990s, software engineering society began to initiate quantitative research on software testability. Software testability study has been a vital research direction since 1990s and became more persistent in 21st century [27, 28].

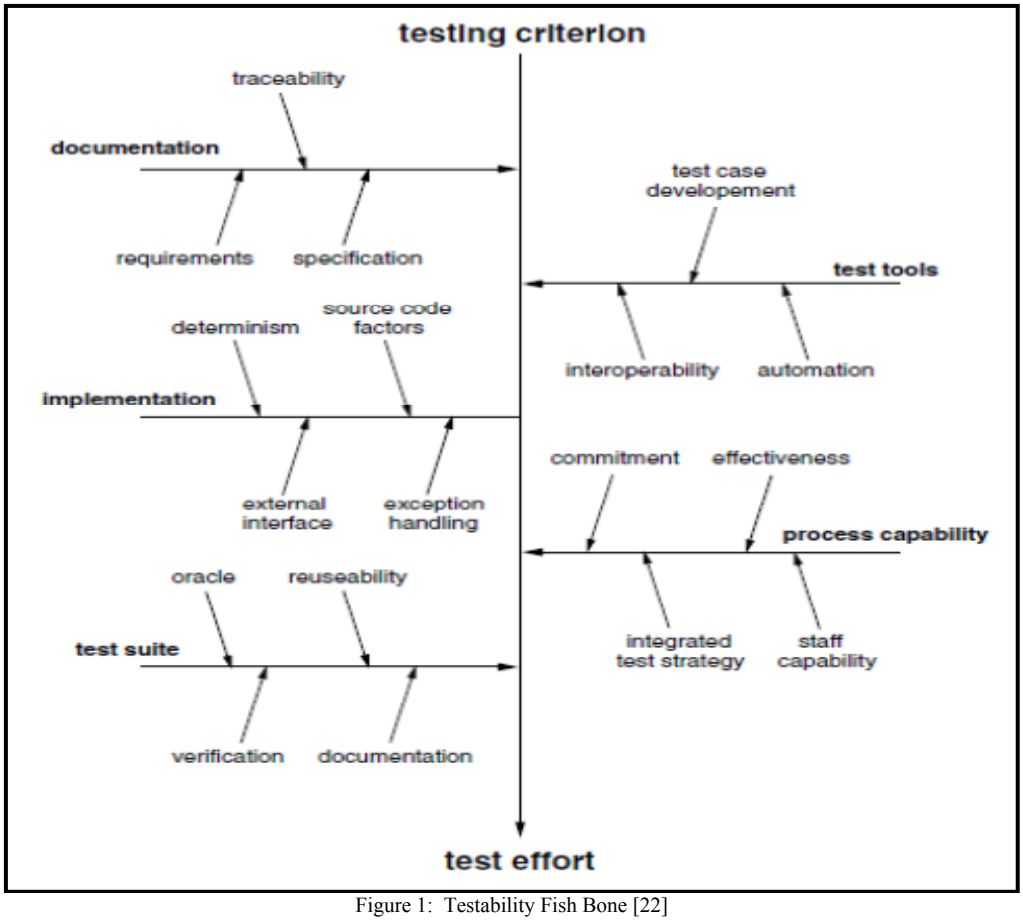

Figure 1: Testability Fish Bone [22]

III. Testability Estimation at Design Face

Practitioners and researchers regularly support that testability should be designed near the beginning in the design stage [18], [19], [20]. The larger part of the studies evaluates testability or more correctly the attributes that have strength on testability but at the source code level. Regardless of the fact that, testability quantification at the source code level is a high indicator of effort estimation, it show the way to the late appearance of information in the development process.

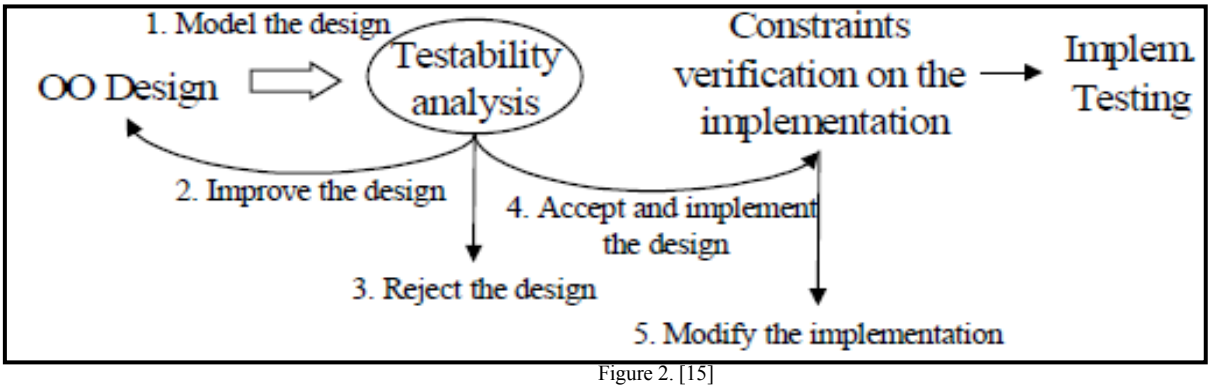

Figure 2. [15]

A result to adjust the design in order to obtain improved testability after coding has started may be really expensive and error-prone [23], [24]. At the same point as estimating testability close to the beginning in the development process appreciably decrease the whole development cost. As an effect, hence, it seems very profitable and important to put into practice testability at the design phase of development life cycle.

TABLE I

S. No.	Authors /	Year	SDLC Phase
01	Abdullah	2014	Design Phase
02	P.Nikfard	2013	Design Phase
03	P.Malla	2012	Design Phase
04	Nazir et al.	2010	Design Phase
05	R A Khan	2009	Design Phase
06	Jerry et al.	2005	Design Phase
07	S.Mouchawrap	2005	Design Analysis
08	Jungmayr	2004	Design Phase
09	Wang	2003	Design Phase
10	Jungmayr	2002	Design Phase
11	Bach	1999	Design Phase
12	Binder	1994	Design Phase
13	J Voas et al.	1992	Design Phase

Testability Estimation at Design Phase consider by various expert [30]

IV. Object Oriented Characteristics

Procedural-oriented languages centre of consideration on procedures, throughout function as the basic unit. You need to first outline all the functions and later than that consider about how to represent data. The object-oriented languages crucial point on components that the user perceives, by means of objects as the essential unit. Object Oriented Programming has vast advantages over other programming styles: The object-oriented technology is much admired in software development environment in modern years. New and more organizations are introducing object oriented system and languages into their software development practices [14].

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. TESTABILITY FACTOR

The mass of the studies evaluate testability or specially the attributes that have impact on testability at the source code level. It has been inferred from the literature survey on testability factors that there is an urgent need of proposing a commonly accepted set of the factors affecting software testability at initial stage of development life cycle [4, 22].

TABLE IIIII

Testability Factors ➔ / Author/Study ➔	Analyzability	Extensibility	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

VI. LITERATURE REVIEW

Wide range of testability evaluation models have been designed in the available literature within last two decades. A quantity of testability methodologies were proposed to assist the designers for measuring and civilizing the testability of object oriented software to generate better-quality and enhanced software systems. Starting from 1975 to 2014, a range of testability estimation models or techniques was developed. At this study we speak about a number of important hypotheses specified by various researches in their work and we will relate those all research through our study in order to promote our work. A variety of studies below present motivation concerning the related work on this area. Binder illustrates software testability as the relative ease and price of revealing software faults i.e., the software sensitivity to faults [7]. Binder offers an exact analysis of the testability factors which are contributing to the software testability evaluation of object oriented design [21], [23]. He declares that testability of object-oriented software, in large sense, is an outcome of six most important factors: individuality of the design, uniqueness of the conclusion, Built-in test abilities, the test suite, test support situation, the software development procedure.

COMPRATIVE STUDY OF VARIOUS APPROACHES

In this phase of study assess the testability models approaches. Study states the contribution of each authors and key concern of every one approach.

S. No.	Authors/ Approach	Year	Donation	Major Issue/Problem
I	Abdullah et al.	2014	♦ Examine empirically the relationship between modifiability and testability of classes diagram at design level. ♦ Perform an empirical validation using object oriented projects and metrics.	♦ Model is too good, but study on big sample of data is considered necessary for better acceptability.
1I	Abdullah et al.	2013	♦ Advocate to measure testability at design phase of development life cycle.	♦ Study required practical implementation
III	Khan et al. [1]	2012	♦ Examine empirically the relationship in the middle of the understandability, complexity model and testability of classes at design level. ♦ Design an empirical study using object oriented artifacts.	♦ Additional study on large sample of data is desirable.
IV	Kout et al. [2] UML	2011	♦ Study empirically the relationship between the model and testability of classes at the source level that design level. ♦ Estimate the ability of the model to forecast testability of classes with using statistical tests.	♦ Not sufficient for Self descriptiveness And both structural and behavioral architecture
V	Khalid et al. [3] UML	2010	♦ broaden the object oriented design metrics ♦ get the proven results ♦ evaluate complexity of design precisely	♦ Accountability ♦ Accessibility
VI	Yogesh Singh et al. [24] UML & Software Contract	2010	♦ Software developers can make utilize of software contracts to reduce the testing attempt. ♦ Software developers can make use of software contracts to improve the testability of the software.	♦ Communicativeness. ♦ Not agreeable for Self-Descriptiveness.

VII	Khan R A & K Mustafa [4] UML	2009	♦ Validate model with structural and functional information ♦ convey the models' ability to estimate overall testability from design information ♦ The model is extra useful in environment having quantitative data on testability ♦ The software developer can apply data to preparation and monitor testing activities	♦ Accountability ♦ Accessibility ♦ Not enough for Self Descriptiveness
			♦ The tester can use testability record to determine on what module to focus during testing	♦ accessibility of built-in test job
VIII	Sharma &Mall[6] UML	2009	♦ Build up a system state model of an object-oriented system from the applicable UML models. ♦ The created developed state model is used to produce test specifications for transition coverage at design level.	♦ Communicativeness. ♦ Not adequate for Self-Descriptiveness.
IX	Zheng & Bundell[7] Test contracts	2008	♦ Software testability quality factors are: traceability, component observability, component controllability, component Understand ability and component test support capability. ♦ advance structure model-based component ♦ Testability at design phase.	♦ Not enough for Self Descriptiveness. ♦ Not enough for both structural and behavioral structural design.
X	Bruntink & Van Deursen[8] Quality model	2006	♦ Maintain quality of the performance with understandable documentation at design era. ♦ Have a preference the reusability and structure of the test suite quality factors. ♦ The assessment of the test support tools used the process capabilities and quality factors. ♦ Factors that manipulate the figure of test cases required for testing	♦ Accountability. ♦ Accessibility.
XI	Mouchawrab et.al[9]	*2005*	♦ They investigated on how to measure testability based on design artifacts at design level ♦ Proposed a framework that may help to estimate testability of design that is mainly modeled with theUML. ♦ Testability investigation at early development stage can yield the highest payoff if focused	♦ Their designs need operational guiding principle on how to maintain in a prearranged and structured manner
XII	Ortega & Rojas[10] Quality model	2003	♦ Express requirements model, design model at design phase, and execution quality model (programming). ♦ The model broaden understanding of the relationship among the attributes (characteristics) and the sub-attributes (sub characteristics) of quality ♦ The quality attributes are maintainability, usability, efficiency, reliability, portability, and functionality.	♦ Accountability. ♦ Critical Accessibility.

XIII	Baudry et al.[11] UML	2002	♦ Construct a model to take into custody class interactions and classify artifact (inheritance and dynamic binding) to evaluation their Cost in terms of number of defined test cases.	♦ The objective of such testing is not clearly stated. ♦ Assumes that various paths between classes are surplus, from a semantic point of view that is costly to test.
XII	Jungmayr et al.[12]	2002	♦ Model relates testability to dependencies between components (e.g., classes) as the more dependencies.	♦ the added tests required to exercise their interfaces
XIV	Voas and Miller[13]	1995	♦ Tells the tester and developer where to give attention to testing effort as this indicates locations in the code where faults could easily hide. ♦ Testing completed as in the early hours as probable that is design point.	♦ The mistake seeding process which can result in a very large number of Executions (high cost) if every possible location for fault seeding is considered.

VII. Important Observations

After victorious completion of the organized literature review a number of important explanations are enumerated as follows.

- Testability evaluation at design phase in the software development life cycle is decidedly suggested by researchers and practitioners.
- In order to quantifying testability of object oriented design study requires to recognize a minimal set of testability factors for object oriented development process.
- Object oriented software characteristics have to be acknowledged and then the set of testability factors appropriate at the design phase should be finalized.
- Further, object oriented testability metrics required to be selected suited at the design phase for the reason that metric selection is a vital step in testability estimation of objects oriented design.

VIII. Conclusion

A group of approaches have been planned in the available literature for quantifying software testability. A review of the suitable literature shows that most efforts have been put at the later phase of software development life cycle. A conclusion to change the design in order to get enhanced testability after coding has started is high expensive and error-prone. For that reason, it is a noticeable truth that quantifying testability near the beginning in the development process significantly reduces overall cost, effort, and rework. On the other hand, the lack of testability at early stage may not be remunerated during succeeding development life cycle.

References

1. 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.
2. Kout, A., Toure, F., & Badri, M. (2011). An empirical analysis of a testability model for object-oriented programs. ACM SIGSOFT Software Engineering Notes, *36*(4), 1. doi:10.1145/1988997.1989020
3. Khalid, S., Zehra, S., & Arif, F. (2010). Analysis of object oriented complexity and testability using object oriented design metrics. Proceedings of the 2010 National Software Engineering Conference on - NSEC '10, 1–8. doi:10.1145/1890810.1890814

4. Khan, R. a., & Mustafa, K. (2009). Metric based testability model for object oriented design (MTMOOD). ACM SIGSOFT Software Engineering Notes, 34(2), 1. doi:10.1145/1507195.1507204

5. Abdullah, Dr, Reena Srivastava, and Dr. M. H. Khan. "Testability Estimation of Object Oriented Design: A Revisit."International Journal of Advanced Research in Computer and Communication Engineering, Volume 2, Issue 8, Pages 3086 -3090, 2013.

6. Sarma, M., & Mall, R. (2009). Automatic generation of test specifications for coverage of system state transitions. Information and Software Technology, *51*(2), 418–432. doi:10.1016/j.infsof.2008.05.002

7. Zheng, W., & Bundell, G. (2008).Contract-Based Software Component Testing with UML Models. Computer Science and its Applications, 2008. CSA '08. International Symposium on, 978-0-7695(13 - 15 October 2008), 83–102.

8. Bruntink, M., & Van Deursen, A. (2006). An empirical study into class testability. Journal of Systems and Software, 79(9), 1219–1232. doi:10.1016/j.jss.2006.02.036

9. Abdullah, Dr, Reena Srivastava, and Dr. M. H. Khan. "Modifiability: A key factor to testability", International Journal of Advanced Information Science and Technology, Volume 26, Issue 26, Pages 62-71,2014

10. [Ortega, M., & Rojas, T. (2003). Construction of a systemic quality model for evaluating a software product. Software Quality Journal, 11:3(July), 219–242.

11. B. Baudry, Y. Le Traon, and G. Sunyé, "Testability Analysis of a UML Class diagram", Proceedings of the Eighth IEEE Symposium on Software Metrics [METRICS.02], IEEE 2002.

12. S. Jungmayr, "Design for Testability", CONQUEST 2002, pp. 57-64.

13. J Voas and Miller , "Improving the software development process using testability research", Proceedings of the 3rd international symposium on software Reliability Engineering, p. 114--121, October, 1992, RTP, NC, Publisher: IEEE Computer Society.

14. Khan R. A. & Nazir M (2007): Testability Quantification of Object Oriented Software: A Critical Review, Proceedings, International Conference on Information & Communication Technology, Dehradun, pp. 960-962., July 26-28, 2007.

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.URL:swerl.tudelft.nl/twiki/pub/Main/ResearchAssignment/RA-Emmanuel-Mulo.pdf

16. Pettichord, B. Design for Testability. In Proc. of Pacific Northwest Software Quality Conference, 2002.

17. Jimenez, G., Taj, S., and Weaver, J. Design for Testability. In Proceedings of the 9th Annual NCIIA Conference, 2005.

18. Jungmayr, S. Testability Quantification and Software Dependencies. In Proceedings of the 12th International Workshop on Software Quantification, pp. 179–202, October 2002.

19. DinoEsposito, "Design Your Classes for Testability", 2008. URL:http://dotnetslackers.com/articles/nnet/Design-Your-Classes-for-Testability.aspx

20. S. Mouchawrab, L. C. Briand, and Y. Labiche, "A quantification framework for object-oriented software testability", Info. And Software Technology, Volume 47, Issue 15, Pages 979-997.December 2005.

21. R A Khan, K Mustafa, S I Ahson, "Software Quality Concepts and Practices", Narosa Publishing House Pvt. Ltd., 2006.

22. S. Mouchawrab et al, "A quantification framework for object-oriented software testability," Carleton University, Technical Report, SCE-05-05, year 2005.

23. R. V. Binder, "Design for Testability in Object-Oriented Systems," Communication of the ACM, vol. 37 (9), pp. 87-101, 1994.

24. Yogesh Singh, Anju Saha ," Prediction of testability using the design metrics for object–oriented software", International Journal of Computer Applications in Technology, Volume 44, Number 1/2012, Pages 12-22, July 2012.

25. Pratima Singh ,Anil Kumar Tripathi," Testing issues" International Journal of Software Engineering & Applications , Issues in Testing of Software with NFR, 3(4), 61 - 76. August 2012.

26. 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.

27. 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

28. J. Gao and Ming-Chih Shih, component testability model for verification and quantification, In Proc. of the 29th Annual International Computer Software and Applications Conference, pages 211–218. IEEE Comp Society 2005.

29. M. Nazir, Khan R A & Mustafa K. (2010): A Metrics Based Model for Understandability Quantification, Journal of Computing, Vol. 2, Issue 4, , pp.90-94. April 2010.

30. 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.

31. Samar Mouchawrab et. al, Carleton University, Technical Report SCE-05-05, 2005

