Volume 4, Issue 10, October 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Testability Quantification of Object Oriented Design: A Revisit

Shahida Khatoon *
R.K.G.IT, Ghaziabad,
India

Dr. Rahul Kumar
I.E.M. Lucknow,
India

***Abstract*— Testability is an elusive concept, its correct measurement or evaluation is a difficult exercise. It is extremely hard to get an understandable view on all the possible factors that have an effect on software testability. Software testability is an external software quality attributes that estimate the complexity and effort required for testing software. The support provided by software testability is important throughout development life cycle and quality assurance.**

The most important concern of this review paper will be systematic study of software testability considering in view by its factors and metrics implementation of testability keeping in mind to supports the testing process and facilitates the creation of better quality software. In this paper primarily study accomplishes a systematic literature review to have extensive facts of testability research and its quality factors and related measurements. Finally study does a comparative analysis on software testability proposed by various experts/researchers including their contribution and limitation. Finally our endeavour is to get the known comprehensive and complete model or framework for evaluating the testability of object oriented design at an initial stage.

Keywords— Software Testability, Testability Quantification, Object Oriented Design Characteristics, Software Quality, Software testing.

I. INTRODUCTION

In today's world, the importance of delivering high quality reliable, testable and maintainable software is no longer an advantage but a necessary factor. Sorry to say, most of the software industries not only fail to produce a quality product to their customers, but in addition do not understand the appropriate quality attributes [1], [25]. With the rising complexity of software applications, software industry are lack to produce the quality software to their customer and even some time some quality attributes are ignored. In present very competitive software market, companies are frequently trying to meet the release dead line that usually reduces the testing time [14]. For this reason the software quality may not be appropriately checked for the probable defects. As a result we cannot take carelessly the quality promise of each software product. Fault prevention and fault recognition have to be considered in every possible phase of development life cycle.

At the present time, testing events are also less than priority so that it turns into easy and effective to find and treat bugs. Software testing is one of the most important activities in the software development life cycle. It is a verification and validation procedure which aims to disclose software faults by executing software product. Software testing is an extremely important means of detecting the software fault [15]. It is too well known truth that more than 50% of the whole software development costs is associated to the software testing activities [22]. For this reason it is one of the most costly phases of software development life cycle in terms of money as well as time. Consequently it is all the time the challenging research area in reducing the cost of testing and producing high quality software within time and budget [16], [17]. Several researchers have paying attention their study for the solutions to reduce the testing cost and effort. If the testability of software can be improved, after that it is possible to decrease the software development cost along with high quality software.

The overall target of software engineering is to produce quality oriented software that is maintainable, testable, committed, and produced inside time, financial plan and as well fulfilled its specific requirements. With number of researches devoting towards the testing phase, a few of the researches have been truly focusing in the direction of the software testability as well. Most important intention will be making the testing procedure effortless and detecting the defects in successful, positive way. With the growing value of testability in software, it is easier to occur the incorrect output in case of presence of defect in the software [18], [19]. The testability approach increases the possibility of revealing the faults finally making software fault recognition 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 met" [28]. 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]. The concept of the software testability given by different authors, it can be in broad-spectrum defined as the degree to which a software artifact make easy procedure of testing. Software testability not simply point out the test process helpfulness but gives new viewpoint on code development [26].

Testability is an elusive concept. It is extremely hard to get an understandable view on all the possible factors that have an effect on software testability. The research on software testability firstly appeared in 1975. It is accepted in McCall and Boehm software quality model, which make the basis of ISO 9126 quality model. From the time when 1990s, software engineering society began to start 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].

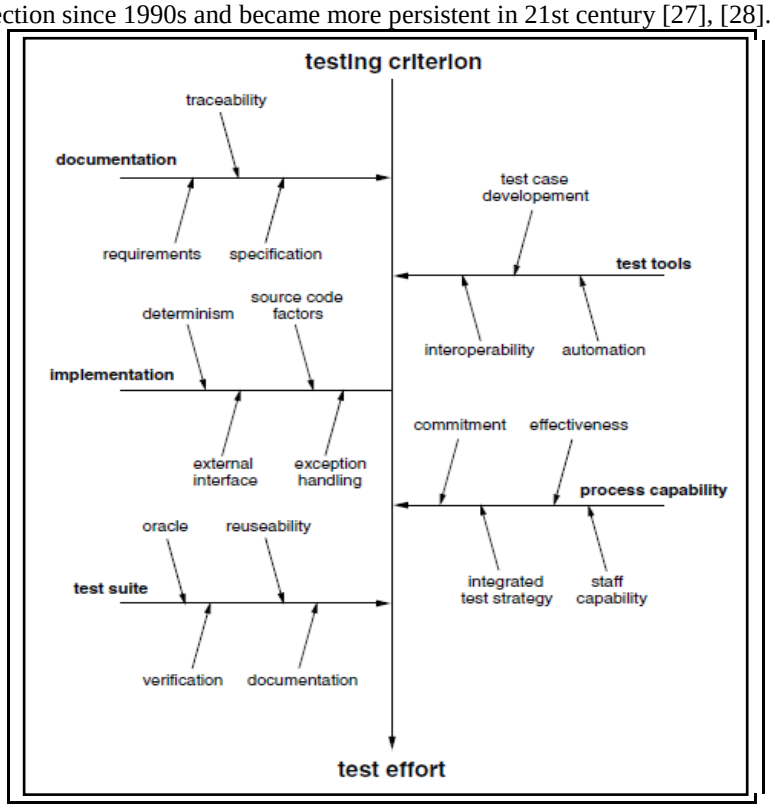

Fig 1: Testability Fish Bone [22]

III. TESTABILITY ESTIMATION AT DESIGN PHASE

Practitioners and researchers frequently advocate that testability should be planned early in the design stage [18], [19], [20]. The greater part of the studies evaluates testability or more accurately the attributes that have force on testability but at the source code level. Despite the fact that, testability quantification at the source code level is a good indicator of effort estimation, it leads to the late appearance of information in the development process. A judgment to modify the design in order to get better testability after coding has started may be extremely expensive and error-prone [23], [24]. At the same time as estimating testability near the beginning in the development process significantly reduce the overall development cost. As an outcome, hence, it seems extremely advantageous and significant to put into practice testability at the design stage of development life cycle.

Table 1

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

Fig 2: Testability Estimation at Design Phase consider by various expert [30]

IV. OBJECT ORIENTED CHARACTERISTICS

Procedural-oriented languages centre of attention on procedures, through function as the basic unit. You require to first figure out all the functions and after that think about how to represent data. The object-oriented languages focal point on components that the user perceives, by means of objects as the fundamental unit. You figure out all the objects by putting all the data and operations that illustrate the user's interaction with the data. Object-Oriented technology has a lot of remuneration:

- *Simplicity in software design* as you could assume in the problem space relatively than the machine's bits and bytes. In OOD you are dealing with high-level concepts and abstractions. Easiness in design direct to more dynamic software development process.
- *No difficulty in software maintenance*: object-oriented software is easier to understand, as a result easier to test, debug, and maintain.

Object Oriented Programming has great advantages over other programming styles: The object-oriented technology is very well-liked in software development atmosphere in recent years. More and more organizations are launching object oriented technique and languages into their software development practices [14]. Object Oriented Software tactics is an approach of structuring software as a group of distinct objects reflecting real-world entities and mapping them into design constructs to characterize relationships and functionality powerfully. It is a sign of an accepted view of the domain and handles inherent complexity improved. Object oriented method presents prospective benefits over traditional software development approach. The function of object oriented technology to systems development has brought a lot of compensation and benefits as well as new demanding issues [29].

The object-oriented technology is more authoritative to design the software in order to make available the product of higher quality. The acceptance of the object-oriented approach is probable to produce improved and cheaper software [1]. Three significant concepts make a distinction the object-oriented approach from conventional software engineering: Coupling, Encapsulation, and Inheritance & Polymorphism [1].These concepts play significant role of design properties in object-oriented software product quality assessment. Finally, show the light on the function of a variety of object oriented design properties such as polymorphism, encapsulation, inheritance, coupling and cohesion on quality attributes such as efficiency, flexibility, understandability and reusability in order to assess the object oriented software product quality.

V. LITERATURE REVIEW

A number of testability theories have been published till date and the testability concept has been grown with different research areas. At this study we talk about a number of the important theories specified by some researches in their paper and we will relate those all research through our thesis in order to encourage our work. Various studies below provide some inspiration regarding the related work on this area. Binder describes software testability as the comparative ease and cost of revealing software faults i.e., the software sensitivity to faults [7]. Binder offers an accurate analysis of the testability factors which are contributing to the software testability estimation of object oriented design [21], [23]. He claims that testability of an object-oriented system, in broad sense, is a result of six most important factors:

- Characteristics of the illustration
- Characteristics of the completion
- Built-in test capabilities
- The test suite
- The test support environment
- The software development process

Binder furthermore listed a few of the testability metrics from encapsulation metric, inheritance metric and polymorphism metric. Encapsulation metric cover up LCOM (Lack of Cohesion in Methods), PAD (Public Access to Data members) even as several of the inheritance metric are NOC (Number of Children),DIT (Depth of Inheritance Tree) and in the same way Polymorphism metric include OVR(Percentage of non-overloaded calls), DYN (Percent of dynamic calls) etc.

COMPRATIVE STUDY OF VARIOUS APPROACHES

In this part study assess the above testability model approaches. Study states the donation of each authors and major issue of every one approach.

Table 2

S. No.	Authors/Approach	Year	Donation	Major Issue/Problem
I	*Khan et al. [1]*	**2012**	♦ Investigate empirically the relationship in the middle of the understand ability, complexity model and testability of classes at design level. ♦ Design an empirical study using object	♦ Further study on large sample of data is needed.

			oriented artifacts.	
II	*Kout et al. [2]* *UML*	2011	♦ Study empirically has the relationship between the model and testability of classes at the source level that design level. ♦ Propose an empirical study by object artifacts. ♦ Estimate the ability of the model to forecast testability of classes with using statistical tests.	♦ Not enough for Self descriptiveness And both structural and behavioral architecture
III	*Khalid et al. [3]* *UML*	2010	♦ broaden the object oriented design metrics ♦ achieve the proven results ♦ calculate complexity of design precisely	♦ Accountability ♦ Accessibility
IV	*Yogesh Singh et al. [24]* *UML & Software Contract*	2010	♦ Software developers can make utilize of software contracts to decrease the testing attempt. ♦ Software developers can make use of software contracts to recover the testability of the software.	♦ Communicativeness. ♦ Not satisfactory for Self-Descriptiveness.
V	*Khan R A & K Mustafa [4]* *UML*	2009	♦ Validate model with structural and functional information ♦ express 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 ♦ Communicativeness ♦ Not enough for Self Descriptiveness
			♦ The tester can utilize testability record to determine on what module to focus during testing	♦ accessibility of built-in test job
VI	*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.
VII	*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 behavioural structural design.
VIII	*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.
IX	*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 modelled with theUML. ♦ Testability investigation at early	♦ Their designs need operational guidelines on how to continue in a organized and structured manner

			development stage can yield the highest payoff if focused	
X	*Ortega & Rojas[10] Quality model*	2003	♦ Demonstrate requirements model, design model at design phase, and execution quality model (programming). ♦ The model enlarge 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.
XI	*Baudry et al.[11]* *UML*	2002	♦ Build up 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 goal of such testing is not evidently stated. ♦ Assumes that numerous paths between classes are redundant, from a semantic point of view that is expensive to test.
XII	*Jungmayr et al.[12]*	2002	♦ Model relates testability to dependencies between components (e.g., classes) as the more dependencies.	♦ the additional tests required to exercise their interfaces
XIII	*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.

VI. IMPORTANT OBSERVATIONS

After successful completion of the organized literature review a number of important observations are enumerated as follows.

- ♦ Testability estimation at design phase in the software development life cycle is highly recommended by researchers and practitioners.
- ♦ In order to quantifying testability of object oriented design study requires to identify a minimal set of testability factors for object oriented development process.
- ♦ Object oriented software characteristics have to be identified and subsequently the set of testability factors relevant 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.

VII. CONCLUSION

A lot of approaches have been planned in the available literature for quantifying software testability. A review of the appropriate literature shows that greatest efforts have been put at the later phase of software development life cycle. A judgment to modify the design in order to get better testability after coding has started is high costly and error-prone. For that reason, it is a noticeable truth that quantifying testability early in the development process greatly reduces overall cost, effort, and rework. On the other hand, the lack of testability at early stage may not be compensated during subsequent development life cycle.

ACKNOWLEDGMENT

I am very much thankful to **Dr. Rahul Kumar** for their valuable suggestions and support.

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] Briand, L. C., Labiche, Y., & He, S. (2009). Automating regression test selection based on UML designs. *Information and Software Technology*, *51*(1), 16–30. doi:10.1016/j.infsof.2008.09.010

[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] Samar Mouchawrab et. Al, Carleton University, Technical Report SCE-05-05, 2005

[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] E. Mulo, "Design for Testability in Software Systems", Master's Thesis, 2007. 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] A. Zaidman et. al, "On how developers test open source software systems", Technical Report TUD-SERG-2007- 012, Delft University of Technology, Software Engineering Research Group, 2007.

[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] Y. Wang, "Design for Test and Software Testability", University of Calgary, 2003. URL:http://www.ucalgary.ca/~ageras/wshop/abstracts/2003/design-for-estability.htm

[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] Mahfuzul Huda, Dr.Y.D.S.Arya, Dr. M. H. Khan: Measuring Testability of Object Oriented Design: A Systematic Review, International Journal of Scientific Engineering and Technology Volume No.3 Issue No.10, pp: 1313-1319, 2014

[31] ISO.International standard ISO/IEC 9126.information technology: Software product evaluation: quality characteristics and guidelines for their use, 1991.

