International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.3 Issue No.10, pp : 1313-1319 1 Oct 2014

Measuring Testability of Object Oriented Design: A Systematic Review

Mahfuzul Huda, Dr.Y.D.S.Arya, Dr. M. H. Khan
Department of Computer Science & Engineering, Invertis University, Bareilly-243123, India
mahfuzul@iul.ac.in

Abstract: Testability is an important quality factor of object oriented software. Its correct measurement or evaluation always facilitates and improves the test process. A lack of testability contributes to a higher level test cost and changeable effort. However testability has always been an indefinable concept and its absolute estimation (measurement) or evaluation is a tough job. Researchers and practioners have always claimed that testability should be considered as key factors which have positive impact on software development in order to ensure the software quality customer satisfaction especially at design face. [29].
The purpose of this review report is to proposing a conceptual comparative evaluation considering the testability issues, limitation and to investigate the general testability factors and commonly accepted minimal set of testability factors with the help of the systematic literature review. In this paper initially we conduct a literature review to have broad knowledge of testability and its quality factors and associated measurements that are found exhibiting the number of different testability factors that are presented by different researchers in different perspective. Next we do a comparative analysis on software testability proposed by various experts/researchers including their contribution and limitation.

Keywords: Software Testability, Testability Estimation, Object Oriented Software, Software Quality, Software testing, Effort estimation.

I. Introduction

Software development processes mainly focus on controlling and reducing errors, Identifying and rectifying software faults that do occur, and support to provide high quality software [26]. It is well understood that delivering quality software is no longer an advantage but is a necessary factor. So we can say that acceptance and success of any software product depends on its quality. The quality can be measured in terms of attributes of the system. Unfortunately, most of the industries not only fail to deliver a quality product to their consumers, but also do not understand the significant quality attributes for ensuring the software quality; testing is the main activity in software development process. Software testing is an important discipline of software engineering, and consumes significant amount of time and effort. An appropriate approach is required to perform testing activities properly and effectively. Software testability always supports the testing process and facilitates the creation of better quality software within given time and budget.

Testability is a quality factor; its measurement or evaluation can be used to predict the amount of effort required for testing and helps allocating required resources. There is no clear definition to ‘what aspects of software are actually related to testability. However, testability has always been an elusive concept and its correct measurement or evaluation is a difficult exercise. Most of the studies measure testability or precisely the quality 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 acute need of proposing a commonly accepted minimal set of the factors affecting software testability [4, 26]. Estimating testability at a later stage leads to the late arrival of desired information, leading to late decisions about changes in design. This greatly increases total cost and rework. Therefore, early evaluation of testability in the development process may enhance quality and reduce testing efforts and costs.

II. Software Testability

The most common definition of Software Testability is ease of performing testing. Software testability is an external software quality attribute that evaluates the complexity and the effort required for software testing. Software testability is a key aspect to allow the detection of difficult error to uncover defects in software. The IEEE Standard Glossary defines testability as the degree to which a system or component facilitates the establishment of test criteria and performance of tests to determine whether those criteria have been met [27].ISO defines it in a similar way: “attributes of software that bear on the effort needed to validate the software product” [28].
Many testability definitions are given by researchers. Binder [16] relates software testability to two properties of the software under test: controllability and observability. To test a component, one must be able to control its input (and internal state) and observe its output (and internal state).Voas et al. defines software testability based on software sensitivity to faults [25]. Briand and Labiche define in] the testability of a model as the degree to which the model has sufficient information to allow automatic generation of test cases[21].Testability is a non-functional requirement important to the testing team members and the users who are involved in user acceptance testing. Non functional requirements are mostly quality requirements and may make the customer satisfied and happy. Software testability is one of the important concepts in design, and testing of software program and components. Building programs and components with high level testability always simplifies test process, reduces total test cost, and increases software quality.
Testability has always been an elusive concept and its correct measurement or evaluation is a difficult exercise because various potential factors have affect on software testability measurement. Testability is one of the most important quality indicators. Most of the studies measure testability or more precisely

The attributes that have impact on testability but at the source code level. However, testability estimation at the source code level is a good indicator of effort estimation; it leads to the late arrival of information in the development process. Estimating testability at later stage of development process after coding has been started may be very expensive and error-prone. But if testability is evaluated earlier in the development process, before coding starts, it may greatly reduce the overall cost, time and rework. As a result it can accelerate the software development process.

III. Testability At Design Phase

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 testability 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 testability should be planned early in the design phase. So it is important to identify object oriented design artifacts to quantify testability measures as early as possible in development life cycle. During identification of design factors which have positive impact on testability estimation, a pragmatic view should be considered. If we consider all factors and measures then they become more complicated, ineffective and time consuming. So need to identify testability factors and measures which affect the activity positively and directly [26]. In order to estimating testability, its direct measures are to be identified. Design level factors like abstraction, encapsulation, inheritance, cohesion, coupling etc. will also be investigated keeping in view their impact on overall testability. This process identifies object oriented design constructs that are used during design phase of development life cycle and serve to define a variety of testability factors. The contribution of each object oriented design characteristics is analyzed for improvement in design testability.

IV. Testability Factors

S. No.	Authors /	Year	SDLC Phase	Ref.
01	Abdullah	2014	Design Phase	I.
02	P.Nikfard	2013	Design Phase	II.
03	P.Malla	2012	Design Phase	III.
04	Nazir et al.	2010	Design Phase	IV.
05	R A Khan	2009	Design Phase	V.
06	Jerry et al.	2005	Design Phase	X
07	S.Mouchawrap	2005	Design Analysis	XXIII
08	Jungmayr	2004	Design Phase	XI
09	Wang	2003	Design Phase	XII
10	Jungmayr	2002	Design Phase	XIV
11	Bach	1999	Design Phase	XV
12	Binder	1994	Design Phase	XVI
13	J Voas et al.	1992	Design Phase	XXV

Table: 1 A Critical Observation Table: consider of Testability Estimation at Design Phase by various Experts / Researchers

The testability of software components (modules and class) is determined by factors that are crucial for an accurate measurement of software quality and reliability .So estimation is based on those factors that can affect software testability directly especially at design phase that is an initial stage of software development life cycle

An accurate measure of software quality and reliability absolutely depends on testability measurement. This is totally based on those factors that can influence software testability directly, especially at design phase. The testability of object oriented software should be evaluated as soon as possible, mainly as it is designed, not when coding is started or completed. Object oriented design characteristics greatly contribute to identify software testability factors that play key role to reduce effort in measuring testability of object oriented design at design phase during software development life cycle. However, testability has always been an elusive concept and its correct measurement or evaluation is a difficult exercise. It is very hard to produce a truthful view on all the factors that have impact in improving testability of object oriented software. It is evident from literature survey that there is a difference of opinion among practitioners in taking into consideration testability factors for estimating software testability of object oriented design in general and at design phase . A consolidated chart for the testability factors identified by various experts is concluded in given Table No.2

It is clearly evident from this Table (No.2) that Observability, Controllability, Changeability, Built-in-test, Reusability, and Understandability are the commonly accepted testability factors at design phase. (Table No.3)

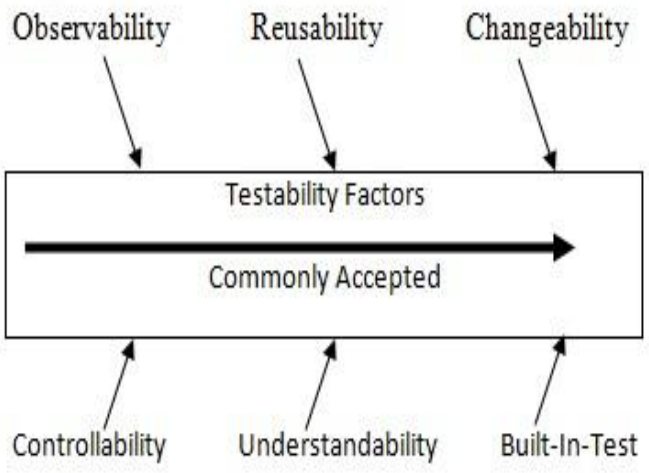

Table No. 3 - Commonly accepted Software Testability Factors at Design Phase

Serial Numbers	Authors/ Experts / Researchers	Year	References	TESTABILITY FACTORS																	
				Observability	Controllability	Built –in – test	Traceability	changeability	Understandability	Modifiability	Reusability	Fault Locality	Simplicity	Complexity	Test Support environment	Test Suite	Development Process	Representation characteristics	Implementation characteristics	Separation of concerns	Coupling
01	Abdullah	2014	I.	✓	✓		✓	✓		✓	✓				✓						
02	P.Nikfard	2013	II.		✓		✓	✓		✓											
03	P.Malla	2012	III.				✓	✓	✓	✓											
04	Nazir et al.	2010	IV.	✓	✓	✓	✓	✓		✓	✓										
05	R A Khan	2009	V.	✓	✓	✓	✓	✓		✓	✓				✓						
06	Dino	2008	VI.		✓		✓	✓		✓											
07	Zheng	2008	VII.	✓	✓		✓		✓												
08	E Mulo	2007	VIII.	✓	✓		✓				✓										
09	Bruntink	2006	IX.	✓				✓			✓					✓					✓
10	Jerry	2005	X.	✓	✓		✓	✓	✓	✓	✓										
11	Jungmayr	2004	XI.	✓			✓													✓	✓
12	Wang	2003	XII.	✓	✓	✓	✓	✓	✓		✓		✓								
13	*Ortega*	2003	XIII.	✓	✓			✓													
14	Jungmayr	2002	XIV.	✓		✓	✓	✓		✓	✓	✓		✓							
15	Bach	1999	XV.		✓			✓	✓	✓			✓			✓					
16	Binder	1994	XVI.	✓	✓	✓	✓	✓								✓	✓	✓	✓		

Table No. 2 : A Critical Observation Table: Testability Factors Consider by Experts / Researchers at Design Phase

V. Comparative Survey - Comparative Analysis of Software Testability by Experts/developers/Researchers/ Practioners

In this section we evaluate and explain the above testability model purposed by various Experts / developers /Researchers/ Practioners also conclude the Contributions and main issue and limitations of each approach.

No.	*Authors* Year Approach	Contributions	Issues & Limitations
1	***Kout et al.***[VII] 2011 UML	➢ Investigate empirically the relationship between the model and testability of classes at the source level that design level. ➢ Design an empirical study using object artifacts. ➢ Evaluate the capability of the model to predict testability of classes with using statistical tests.	➢ Bounded accessibility ➢ Not sufficient for both structural and behavioral architecture
2	***Khalid et al.***[XVIII] 2010 UML	➢ Use design phase to extend the object oriented design metrics. ➢ Obtain the quantifiable results. ➢ Predict complexity of design accurately.	➢ Complex Accessibility. ➢ Not sufficient for Self-Descriptiveness.

3	***Yogesh Singh et al.***[XIX] 2010 **UML & Software contract**	➢ Software contracts improve the testability of an object oriented class that reduces the testing effort up to 50% to test a class at design time. ➢ Software developers can make use of software contracts to reduce the testing effort. ➢ Software developers can make use of software contracts to improve the testability of the software.	➢ Accountability. ➢ Accessibility. ➢ Communicativeness. ➢ Not sufficient for Self-Descriptiveness.
4	***Khan R A & K Mustafa***[V] 2009 UML	➢ Testability model Validate using structural and functional information. ➢ Demonstrate the models' importance to evaluate overall testability from design phase information. ➢ The model is more practical in nature having quantitative data on testability. ➢ The researchers/tester can use testability information to determine on what module to focus during testing.	➢ Less sufficient for Self-Descriptiveness.
5	***Sharma &Mall***[XXI] 2009 UML	➢ Develop a system state model of an object-oriented system from the relevant UML models. ➢ The synthesized developed state model is used to generate test specifications for transition coverage at design level.	➢ Communicativeness. ➢ Not sufficient for Self-Descriptiveness.
6	***Briand***[XX] 2009 UML	➢ Implements the regression test selection problem at the design level in the context of UML-based development. ➢ Higher efficiency in test selection based on the design change analysis and changeability between UML designs and Regression test cases. ➢ Better for regression test effort earlier in the change process (design phase) that is once design changes have been determined.	➢ Limited Accountability. ➢ Accessibility depends upon single attribute.
7	***Zheng & Bundell***[XXII] 2008 Test contracts	➢ Testability quality factors are: traceability, component observability, component controllability, component Understandability and component test support capability. ➢ Improve structure model-based component ➢ Testability at design level.	➢ Not sufficient for Self-Descriptiveness. ➢ Not sufficient for both structural and behavioral architecture.
8	***Bruntink & Van Deursen***[IX] 2006 Quality model	➢ Support quality of the implementation with clear documentation at design time. ➢ Prefer the reusability and structure of the test suite quality factors. ➢ The estimation of the test support tools used the process capabilities and quality factors. ➢ Factors that influence the number of test cases required testing	➢ Accountability bound. ➢ Accessibility. ➢ Not sufficient for both structural and behavioral architecture.
9	***Mouchawrab et.al***[XXIII] ***2005***	➢ They investigated on how to measure testability based on design artifacts at design level and proposed a framework that may help to assess testability of design that is particularly modeled with the UML. ➢ Testability analysis at initial development stage can yield the highest payoff if focused (during analysis and design stages of object-oriented development.).	➢ Their designs lack operational guidelines on how to proceed in a systematic and structured manner.

11	***Ortega & Rojas***[XIII] 2003 Quality model	➢ Illustrate requirements model, design model at design phase, and implementation quality model (programming). ➢ The model increase understanding of the relationship between the attributes (characteristics) and the sub-attributes (sub-characteristics) of quality ➢ The quality attributes are: efficiency, reliability, maintainability, portability, usability and functionality.	➢ Accountability. ➢ Critical Accessibility. ➢ Not sufficient for both structural and behavioral architecture.
12	**Baudry et al.**[XXIV] 2002 UML	➢ Develop a model to capture class interactions And define artifact (inheritance and dynamic binding) to evaluation their Cost in terms of number of defined test cases. At initial stage to be.	➢ The objective of such testing is not clearly stated. ➢ Assumes that multiple paths between classes are redundant, from a semantic viewpoint that is expensive to test.
13	**Jungmayr *et al.***[XIV]2002	➢ Model relates testability to dependencies between components (e.g., classes) as the more dependencies.	➢ the more tests required to exercise their interfaces
14	**Voas and Miller**[XXV] 1995 [25]	➢ Tells the tester and developer where to Concentrate testing effort as this indicates locations in the code where faults could easily hide. ➢ Testing done as early as possible that is design time.	➢ The fault seeding procedure which can result in a very large number of Executions (high cost) if every possible location for fault seeding is considered.
15	**Binder *et al.***[XVI] 1994	➢ A more testable system provides ➢ Increased reliability for a fixed testing budget. ➢ Software testability is a result of six high-level factors: 1 built-in test., 2 test suite, support environment, implementation characteristics, and representation characteristics at initial phase.	➢ Does not provide any Empirical evidence that there is a correlation between the suggested metrics and testability.
16	**Freedman *et al.*** 1991	➢ Provide domain testability at design level as the ease of modifying a program so that it is observable and controllable. ➢ Defines that a software artifacts that is easily test-able has the desirable quality attributes: test sets are non-redundant, test sets are small, test outputs are easily interpreted and software faults are easily findable.	➢ It does not exhibit any test input-output inconsistencies
17	**Bache and Mullerburg**[xv] 1990 quality assurance	➢ Use control-flow graphs to relate testability to the effort needed for testing, and estimating testability as the minimum number of test cases of a Given coverage criteria. ➢ Assuming full coverage is possible with Fenton-Whitty theory at design time. .	➢ The approach is however Limited to control-flow based testing strategies, although, as acknowledged by the authors. ➢ Testability measurement is also dependent on the selected coverage criterion.

VI. CRITICAL OBSERVATIONS

After successful completion of the systematic literature review some important. Critical observations are as follows.

- An efficient and accurate estimation of software testability at an initial stage that is design phase in the software development process is highly recommended by researchers and practitioners.
- Estimation of software testability at design phase may greatly improve the software quality, user satisfaction, and reduce effort of rework.
- For minimizing effort in measuring testability of object oriented design, one needs to identify a minimal set of commonly accepted set of the testability factors early in design phase for object oriented development process, which have positive impact on testability estimation.
- Object oriented software artifacts must be identified and then the minimal set of testability factors relevant at the design phase should be finalized.

VII. CONCLUSION

- Numbers of different approaches have been proposed in the systematic literature review for measuring software testability.
- Numbers of different software testability quality factors are presented in different researches in different perspectives by practioners /researchers.

A survey review of the relevant literature proves that maximum efforts are being put at the later phase of software development life cycle. So I can say that decision to change the design in order to improve software testability after coding has started, is very expensive and error-prone. Therefore, it is an obvious fact that measuring testability early in the development process (design process) may greatly reduce testing time, effort, rework and cost. The early estimation of testability at design phase can yield the highest payoffs.

ACKNOWLEDGEMENT

I am very much thankful to **Dr. Y.D.S. Arya Sir and Dr. M. H. Khan Sir** for their valuable suggestions and support.

REFERENCES

i. *Abdullah, Dr. Reena Srivastava, Dr. M.H. Khan International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682 Vol.26, No.26, June 2014*

ii. *Pourya Nikfard,An Empirical Study into Model Testability*

iii. *P.Malla ,An Analysis on Software Testability and Security in Context of Object and Aspect Oriented Software Development,2012*

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

v. *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*

vi. *Dino Esposito, "Design Your Classes for Testability", 2008. URL:http://dotnetslackers.com/articles/net/Design-Your-Classes-for-Testability.aspx*

vii. *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.*

viii. *E. Mulo, "Design for Testability in Software Systems", Master's Thesis, 2007.*

ix. *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*

x. *A Component Testability Model for Verification and Measurement ,URL: http://doi.ieeecomputersociety.org/10.1109/COMPSAC.2005.17*

xi. *S. Jungmayr, "Improving testability of object-oriented systems", Master's Thesis, 2004, ISBN 3-89825-781-9. URL: http://www.dissertation.de*

xii. *Y. Wang, "Design for Test and Software Testability", University of Calgary, 2003.*

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

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

xv. *Bach, James."Heuristics of Software Testability" (1999).*

xvi. *Binder, Robert V. "Design for testability in object-oriented systems." Communications of the ACM 37.9 (1994): 87-101.*

xvii. *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*

xviii. *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*

xix. *Singh, Y., & Saha, A. (2010). Improving the testability of object oriented software through software contracts. ACM SIGSOFT Software Engineering Notes, 35(1), 1. doi:10.1145/1668862.1668869*

xx. *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*

xxi. *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*

xxii. *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.*

xxiii. *Samar Mouchawrab et. Al ,Carleton University, Technical Report SCE-05-05 ,2005*

xxiv. *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.*

xxv. *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.*

xxvi. *Abdullah, Dr. Reena Srivastava, Dr. M. H. Khan "Testability Estimation of Object Oriented Design: A Revisit", in IJARCCE, Volume 2, Issue 8, Pages 3086-3090, August 2013.*

xxvii. *IEEE Press, "IEEE Standard Glossary of Software Engineering Technology,"ANSI/IEEE Standard 610.12-1990, 1990*

xxviii. *International standard ISO/IEC 9126. Information technology: Software Product Evaluation: Quality Characteristics and Guidelines for Their Use., 1991*

xxix. *Mahfuzul Huda Abdullah. Effort Estimation in Measuring Testability of Object Oriented Design URL: http://conference.aimt.edu.in/csit/*

Technical Biographies

Mahfuzul Huda currently pursuing Ph.D. in the field of software engineering of Department Computer Science and Engineering from, **Invertis University, Bareilly, India**.

Dr. Y.D.S. Arya is currently working as **Pro - Vice Chancellor, Invertis University Bareilly, India, Dr. Arya** obtained M.Tech. Degree in Computer Science and Engineering from IIT Kanpur before his Ph.D. in Computer Science and

Engineering. He has more than 30 years of experience in teaching and programming. He has developed few software packages and has contributed in the development of Solaris Kernel 9.0. Apart from working at IIT Kanpur, he has worked with Sun Microsystems, San Francisco, Ca, USA and Fujitsu at Numazu Company in Japan.

Dr. M. H. Khan Associate Professor, Department of Computer Science and Engineering at IET Lucknow UP. Obtained his MCA degree from Aligarh Muslim University (Central University) in 1991 .Later he did his PhD from Lucknow University. He has around 25 years rich teaching experience at UG and PG level. His area of research is Software Engineering. Dr. Khan published numerous articles, several papers in the National and International Journals and conference proceeding.