ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

Testability Estimation of Object Oriented Design: A Revisit

Abdullah[1], Dr. Reena Srivastava[2], Dr. M.H. Khan[3]

Research Scholar, School of Computer Application, BBDU, Lucknow, India [1]

Dean, School of Computer Application, BBDU, Lucknow, India [2]

Associate Professor, Department of C.S. E., I.E.T., Lucknow, India [3]

Abstract: Testability is one of the most important quality indicators. Its correct measurement or evaluation, always facilitate and improve the test process. However, testability has always been an elusive concept and its correct measurement or evaluation is a difficult exercise. Researchers and practitioners have always argued that testability should be considered as a key attribute in order to guarantee the software quality. An accurate measure of software quality depends on testability measurement, and as a result estimating efforts in measuring testability is a complex problem attracting considerable research attention.

This paper presents the results of a systematic review conducted to collect evidence on software testability estimation of object oriented design. In this review paper, our aim is to find the existing known comprehensive and complete model or framework for evaluating the testability of object oriented design at an initial stage.

Keywords: Software Testability, Testability Estimation, Object Oriented Design, Software Quality, Software testing.

I. INTRODUCTION

Software development processes mainly focus on minimizing errors, detecting and correcting software faults that do occur, and help to deliver high quality software after development. It is well understood that delivering quality software is no longer an advantage but is a necessary factor. Unfortunately, most of the industries not only fail to deliver a quality product to their customers, but also do not understand the relevant quality attributes [1]. 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 time and budget.

Effective testability planning, early in the software development process may greatly contribute to the delivery of high quality software products; more satisfied and happy users, highly reduced overall maintenance cost and rework, and more accurate and reliable results. However, ineffective testability planning or testability planning at later stage of development process will lead to the opposite results; lower quality products, unsatisfied users, increased maintenance cost, unreliable and inaccurate results. Software testing, which can expose software faults during development, is an important means for software quality assurance. With the enlargement of software scale and complexity, testing problems have become more prominent. For example, money needed in testing is regularly increasing, and more and more tests are difficult to carry out. A major means to solve these problems is by improving the software testability. Software testability estimation can help developing a more reliable software by more thoroughly understanding the software testability [13].

Testability is a quality factor; its measurement or evaluation can be used to predict the amount of effort required for testing and help in allocating required resources. There is no clear definition to 'what aspects of software are actually related to testability' [17]. 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 attributes that have impact on testability at the source code level. It has been inferred from the literature survey on testability analysis that there is a heavy need of identifying a commonly accepted set of the factors affecting software testability [18]. Estimating testability at a later stage leads to the late arrival of desired information, leading to late decisions about changes in design. This simply increases cost and rework. Therefore, early estimation of testability in the development process may greatly enhance software quality and reduce testing efforts and costs. Hence measuring software testability at an early stage of development life cycle may reduce the overall cost, cycle time and rework.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

II. SOFTWARE TESTABILITY

An accurate measure of software quality depends on testability measurement [17]. 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 make the customer happy and satisfied. 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. Software testability analysis may be useful to examine, and estimate the quality of software. Testability is important for both ad-hoc developers, and organizations with a high-level of process maturity. It reduces cost in a reliability-driven process, and increases reliability in resource-limited processes. It refers to 'the inherent ability, or extent of ease with which software undergoes through testing [27]. Any tool or technique that help to improve object oriented design at an initial stage of software development life cycle can have highly appreciable impact on the final testing cost and 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. Its measurement leads to the prospects of facilitating and improving a test process. Several approaches, prominently including the Program-based Testability Measurement, Model-based Testability Measurement, and Dependability-based Testability Assessment have been proposed [15]. How efficiently the faults will be uncovered depends upon the testability of the software [16]. 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 ESTIMATION OF OBJECT ORIENTED DESIGN

Object oriented technology have become the most popular and familiar concept in software industry. Object oriented concept is now widely used by software industry. Despite the fact that technology is not mature enough from testing point of view [17], almost everyone talk about it, almost everyone claim to be doing it and almost everyone says that it is better than traditional function oriented design. Because most of the focus of the object oriented approach to software development has been on analysis and design phase, only a few research studies have been devoted to explore the concept of testability in object oriented system.

Practitioners frequently advocate that testability should be planned early in the design phase. So it is necessary to identify object oriented design artifacts to quantify testability measures. 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 or time consuming. Therefore, there is a need to identify factors and measures which affect the activity positively and directly. 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 identified 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. LITERATURE REVIEW

Testability is an elusive concept. It is very difficult to get a clear view on all the potential factors that can affect software testability. The research on software testability first appeared in 1975. It is adopted in McCall and Boehm software quality model, which build the foundation of ISO 9126 quality model. Since 1990s, software engineering community began to launch quantitative research on testability. Software testability analysis has been an important research direction since 1990s and became more pervasive in 21st century [1].

A number of researchers addressed software testability, but in the context of conventional structured design. The question of testability has been revived with the object-orientation [2, 3, 4]. Despite the fact that object oriented technology has now been widely accepted by the software industry, only a few research studies have been devoted to explore the concepts of testability in object oriented systems. Several developments on the measurement of testability and design for testability have been reported in the literature [9, 10, 16, 17, and 18]. Unfortunately, these achievements have not been widely accepted and hence, not been adopted in practice by industry [1, 5]. Following sections systematically summarize some of the relevant important efforts made by researchers in this area.

In 1993, Voas and Miller [6] highlight software testability metric that are depends upon inputs and outputs artifacts of a software module. To measure testability, they proposed PIE (propagation, infection and execution) analysis

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

technique [2]; but measuring testability through the PIE analysis technique was very complex and have high complexity. Due to these limitations this technique is not adopted by industry personals.

In 1994, Binder had done a great work showing the importance for improving software testability in system development life cycle [4]. He proposed a fishbone model representing the key factors of testability. Fishbone model broadly include, software testability is a result of six factors: (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. But unfortunately all above testability factors measure only higher level of abstraction. Which result has no any clear relationship with object oriented design constructs and implementation.

In 1998, Bruce and Haifeng Shi [7] highlighted the object oriented design testability factors that affect testability, and showing their impact for improving software testability of object oriented design. They designed a model for testability measurement with the help of single testability factor and design level constructs. But this framework has various limitations from implementation point of view.

In 2002, Jungmayr [8] showing a new concept for estimating software testability through integration testing and emphasized only these component. He identified local dependencies that positively contributing and responsible for overall testability. Jungmayr's concept *used* reduction metric to calculate the effect of individual factors in software testability to find out required testability metric.

In 2004, *Bruntink and van Deursen* [9] presents a group of metrics for exploring the testability of object oriented Java system, and identified testability factors through source code metrics. In order to improve testability, a decision to change the design after coding was started it very expensive and error prone.

In 2005, Baudry et al. highlighted the importance of individual types of class coupling on testability metrics. And establish a relation of coupling and class interaction metrics that finalize testability [11]. However, this hypothesis was not empirically validated.

In 2005, for measuring testability, Jerry and Ming presented a model that is based on pentagon shaped and analytical approach [10].

In 2007, Mulo integrate the importance of testability measurement throughout the software development life cycle [12]. Estimating testability during the entire development life cycle is very expensive and error prone.

In 2009, Jianping Fu & Minyan Lu, proposed a request-oriented method of software testability measurement [19]. The method can select suitable elements from a self-contained software testability measurement framework according to the different measurement requests to complete all kinds of software testability measurement. Unfortunately, these achievements have not been widely accepted and hence, has not been adopted in practice by the industry.

In 2011, Fadel Toure [20] presented a novel approach for improving testability of the software using software reliability growth models. However, the different approaches discussed for improving and measuring testability were theoretical and the quantitative measure of improvement of testability was not given.

In 2012, Badri, Mourad, and Fadel Toure, focused on establishing the relationship between object oriented metrics and testability of classes in terms of required testing effort [21]. For this they performed an empirical analysis using data collected from three Java software systems for which JUnit test cases exist. To capture testability of classes, they used different metrics to quantify the corresponding JUnit test cases. The metrics related to the JUnit test cases were used, in fact, to classify 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. [22] proposed fuzzy classification approach. In this approach a large collection of classifiers is available with subsets of the software metric features. Subsets are selected stochastically using a fuzzy logic based sampling method. The classifiers then predict the quality, specifically the class label, of each software object. Fuzzy integration is applied to the results from the most accurate individual classifiers. These classifiers estimate the quality especially at the class level, which lead to no clear relationship with the software metric features that are based on design artifacts and the implementation level.

In 2013, Tiwari, Rajeev and Noopur Goel [23] proposed reuse-oriented test approaches, which are used to reduce the testing effort. Further, he stated the state-of-the-art in reuse-oriented test approaches employed in reuse oriented development processes. But this approach is not widely acceptable for new products. The reuse-oriented approach is not always practical in its pure form because a full repertoire of reusable components may not be available

In 2013, Panigrahi, Chhabi Rani, and Rajib Mall [24] proposed a regression test case prioritization technique for object-oriented programs. They construct an intermediate graph model of a program from its source code. When the program is revised, the model is updated to reflect the changes. The constructed model shows control and data dependencies, and information pertaining to various types of dependencies arising from object-relations such as association, inheritance and aggregation. Regression test has been investigated from the perspective of source code. It ensures that old code still works when the new code changes are done. This is practically very expensive, wasteful of resources, and requires unduly long times.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

In 2013, Amid, Amin, and Somaye Moradi [25] proposed a model in order to measure the quality of Software. In this model an effort has been put to increase software productions quality, the process of software production has been determined, using CMM standard framework of maturity level. But in CMM it does not present a method for measurement and evaluation maturity level. They proposed a hybrid model of CMM and COBIT by considering the factors affect on software quality. The factors that influence the software quality include customer view and project management view. However, the hypothesis was not empirically validated and not adopted by industry. In 2013, Kaur, Kiranjit, and Sami Anand [26] proposed a multivariate linear model, which estimates the maintainability of a class diagram. He mainly focused only on maintainability estimation in terms of testability, reliability, portability. These metrics help a software designer for improving the maintainability of a class diagram in the design phase. After a revision tour we found that various methods or techniques are available in the literature for estimating software testability. A survey of the testability estimation of object oriented design shows that maximum effort focus at the later stage of software development life cycle. On the other hand, recent developments in software configuration management frequently advocate integrating software testability in design phase which in turn will help the designer to improve quality of software and security and greatly reduce the overall cost and rework.

V. CRITICAL OBSERVATIONS

After successful completion of the literature survey some important observations can be enumerated as follows.

- If we estimate the software testability at an initial stage that is design phase in the software development process may greatly improve the software quality and as well as client satisfaction, and reduce overall cost, time and effort of rework.
- In order to reducing effort in measuring testability of object oriented design we need to identify a minimal set of testability factors for object oriented development process, which have positive impact on testability measurement.
- Object oriented software characteristics must be identified and then the set of testability factors relevant at the design phase should be finalized.
- Further, testability metrics must be selected at the design phase because metric selection is an important step in testability estimation of objects oriented design.

VI. CONCLUSION

Several approaches have been proposed in the literature for measuring software testability. A survey of the relevant literature shows that maximum efforts have been put at the later stage of software development life cycle. A decision to change the design in order to improve testability after coding has started, but is very expensive and error-prone. Therefore, it is an obvious fact that estimating testability early in the development process may greatly reduce testing time, effort, rework and cost. The early estimation of testability at design phase can yield the highest payoffs. On the other hand, the lack of testability at early stage may not be compensated during subsequent development life cycle.

After the above discussion our conclusion is that testability is a quality factor that attempts to predict that how much effort will be required for software testing. After an exhaustive review process we found that reducing effort in measuring testability of object oriented design is must in order to deliver quality software within time and budget.

REFERENCES

[1] 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.
[2] Voas and Miller, "Software Testability: The New Verification". IEEE Software. Vol. 12(3), p. 17-28, 1995.
[3] J.M. Voas. "Object-Oriented Software Testability". In proceedings of *International Conference on Achieving Quality in Software*, January 1996
[4] R.V. Binder, "Design for testability in object-oriented systems". Communications of the ACM. Vol. 37(9), p. 87-101, 1994.
[5] 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*
[6] Voas and Miller, Semantic metrics for software testability, *Journal of Systems and Software*, Vol. 20 (3), pp. 207-216, 1993.
[7] 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.
[8] Jungmayr, S. Testability Measurement and Software Dependencies. In Proceedings of the 12th International Workshop on Software Measurement, pp. 179–202, October 2002.
[9] 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.
[10] 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.
[11] Baudry and Traon, Measuring Design Testability of a UML Class Diagram. Information and Software Technology, 47(13):859–879, 2005.
[12] E. Mulo, "Design for Testability in Software Systems", Master's Thesis, 2007.
URL:swerl.tudelft.nl/twiki/pub/Main/ResearchAssignment/RA-Emmanuel-Mulo.pdf
[13] Jianping, Fu, Liu Bin, and Lu Minyan."Present and future of software testability analysis." *Computer Application and System Modelling (ICCASM), 2010 International Conference on*. Vol. 15. IEEE, 2010.
[14] D. Esposito, "Design Your Classes for Testability", 2008. http://dotnetslackers.com/articles/n net/Design-Your-Classes-for-Testability.aspx
[15] 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.
[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.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 8, August 2013

[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21]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.
[22] Pizzi, Nick J. "A Fuzzy Classifier Approach to Estimating Software Quality." *Information Sciences* (2013). Volume 241, Pages 1–11, 20 August 2013.
[23]Tiwari, Rajeev, and Noopur Goel. "Reuse: reducing test effort." *ACM SIGSOFT Software Engineering Notes* 38, no. 2: 1-11, 2013.
[24] 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.
[25] Amid, Amin, and Somaye Moradi. "A Hybrid Evaluation Framework of CMM and COBIT for Improving the Software Development Quality." 2013.
[26] 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.
[27]http://developeriq.in/articles/2008/jan/12/software testability

BIOGRAPHY

Abdullah received the MCA degree from Uttar Pradesh Technical University, Lucknow, in 2006. He is currently working as an Assistant Professor, in the Department of Computer Application, at Institute of Environment and Management, Lucknow. His research interests Include Software testability, Software Quality Estimation. He has written various books and study materials for North Orissa University, Suresh Gyan Vihar University, Jaipur, Rajasthan.

Dr.Reena Srivastava is currently working as Dean, School of Computer Applications at BBD University. She received her Ph.D degree from MNNIT Allahabad, India. Her research area includes Multi-Relational Classification, Privacy Preserving Data Mining and Software Engineering.

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 24 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 proceedings.