ISSN: 2321-7782 (Online)
Volume 3, Issue 4, April 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Testability Quantification of Object Oriented Software: A New Perspective

Shahida Khatoon[1]
M.Tech (CS)
RKGIT, Ghaziabad, India

Dr. Rahul Kumar[2]
I.E.M. Lucknow,
India

Abstract: Quantifying testability near the beginning in the software development life cycle mainly at design stage play a vital roles in producing high quality software. On the other hand, most of the mechanism available for testability Quantification may be used in the later phases of development life cycle. Practitioners repeatedly advocate that testability should be planned early in development life cycle. Testability Quantification early in design phase is very much emphasized in this paper. As an outcome, it comprehensively reduces rework during and after implementation. An endeavor has been put forth in this paper to identify the testability key factors contributing in testability Quantification. In addition, testability Quantification model is formulated to quantify software testability in design phase. The formulated model has been validated using experimental test. In conclusion, study incorporates the empirical validation of the testability model as the author's most significant contribution.

Keywords: Object Oriented Metrics, Testability Factors, Quantification, Design Phase, Development Life Cycle.

I. Introduction

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 during design, coding, testing, and quality assurance [1]. However, testability is an elusive concept, its correct quantification or evaluation a difficult exercise. Furthermore, it is not easy to obtain a clear view on all the possible factors that have an effect on testability and the leading degree of these factors under different testing perspectives [3]. Testability estimation 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 choice to change the design in order to improve testability after coding has started may be very expensive and error-prone. Despite the fact that estimating testability early in the development process may significantly reduce the overall cost. This may shape a road-map to industry personnel and researchers to assess, and preferably, quantify and improve software testability at design phase [5, 6, 7]. So reducing effort and improving software testability is a key objective in order to reduce the number of defects that result from poorly designed software.

II. Software Testability Factors

The majority of the mechanisms available for testability evaluation of objects oriented software usually are used in later phases of system development life cycle. Such methods provide a signal of testability and hence quality, but too late to improve the product, prior to its completion [8, 9]. For this reason, it is hard to get an understandable view on all the prospective factors that can affect testability at design phase of development life cycle [10]. Following section in brief present some of the appropriate efforts and donations made by researchers and practitioners in the way of finding the testability factors. Robert Binder, a testability expert did a novel effort in finding out the necessity and significance of testability for developing quality software system [11, 21, 22]. He believed six factors of testability including representation characteristics, implementation characteristics, built-in-test capabilities, test suite, test support environment, development process [14]. All the above factors are illustrated at a high level of constructs most important to no clear correlation with the metrics that are based on design artifacts and the implementation.

The above mentioned explanation reveals that there is a disagreement among practitioners in taking into consideration the factors while estimating testability in common and exclusively at design level. Therefore, it seems extremely desirable to categorize the factors affecting testability. Despite the fact that, receiving a universally accepted set of testability factor is not possible [17, 18, 20], an effort has been made to assemble a commonly accepted set of factors that can affect testability. However, without any loss of generality, it appears realistic to take in the factors namely 'Understandability, reuseability, understandability, and analyzability as testability factors.

III. Object Oriented Design Constructs

Procedural-oriented languages center 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 [12]. 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 benefits:

» *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. 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 [19]. 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.

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 [15]. Three significant concepts make a distinction the object-oriented approach from conventional software engineering: Coupling, Encapsulation, and Inheritance & Polymorphism [8, 22].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 design properties such as encapsulation, inheritance, polymorphism, coupling and cohesion on quality attributes such as efficiency, understandability, analyzability and reusability in order to assess the object oriented software product quality.

IV. Major Contributors

On or after the above mentioned conversation it is uncovered that the set of factors contributing testability quantification varies with the dissimilar discipline of opinions. There is no comprehensive set of testability factors to be addressed while quantifying testability. However in section 3, it was identified that 'a key factors of software testability take in analyzability and understandability'. It is in advance observed that both of the factors have its own involvement for quantifying testability of OO software in design phase. The only motivation to found key factors of testability is to make the designer understand the factors to be addressed while taking into account testability. Consequently it is equally compulsory to address the recognized key testability factors to incorporate testability during development cycle. However it is also very significant to know the weightage of the factor to be addressed and its contribution in making testable software. A factor with low weightage need not be addressed because of its insignificant contribution in testability estimation at design phase.

V. Models Development

In order to set up all the two models following multivariate linear model (1) has selected.

$Y=a_0+a1X1+a2X2+a3X3+\text{-------}+anXn$ ***Eq.* (1)**

Where,

- **Y** is dependent variable.
- X1, X2, X3--------Xn are independent variables.
- a1, a2, a3--------an., are the coefficient.
- And a_0 is the intercept.

VI. Analyzability Quantification Model

In order to establish a model for Analyzability of class diagram, metrics listed in [16], will play the role of independent variables while Analyzability will be taken as dependent variable. The data used for developing Analyzability model is taken from [16]. Using SPSS, values of coefficient are calculated and Analyzability model is formulated as given below.

Analyzability= .359 + .025 × Encapsulation + .058 × Coupling + .044 × Cohesion + .251 × Inheritance ***Eq.* (2)**

TABLE I

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.994[a]	.989	.967	.36940

a. Predictors: (Constant), Inheritance, Coupling, Cohesion, Encapsulation

VII. Understandability Quantification Model

In order to establish a model for understandability of class diagram, metrics listed in [16], will play the role of independent variables while analyzability will be taken as dependent variable. The data used for developing analyzability model is taken from [16]. Using SPSS, values of coefficient are calculated and understandability model is formulated as given below.

Understandability = .795 + .324 × **Coupling** + .148 × **Cohesion** -.036 × **Encapsulation** ***Eq.* (3)**

TABLE II

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.967[a]	.935	.886	.510

a. Predictors: (Constant), Encapsulation, Coupling, Cohesion

VIII. Testability Quantification Model Development

The generic quality models [2] [4] [13] have been taken as a foundation to develop the Testability Quantification Model for Object Oriented Design (TQM^{OOD}). In order to set up a model for testability quantification, a multiple linear regression method has been used to get the coefficients. Applying this method, Analyzability Model [2], Understandability Model [3] have been developed that are given below in equation (4) and (5) respectively.

Analyzability= .359 + .025 × **Encapsulation** + .058 × **Coupling** + .044 × **Cohesion** + .251 × **Inheritance** ***Eq.* (4)**

Understandability = .795 + .324 × **Coupling** + .148 × **Cohesion** -.036 × **Encapsulation** ***Eq.* (5)**

Taking into account the testability contributors 'analyzability and understandability', set up a model for Testability Quantification.

The data used for developing model given in equation (6) has been taken from [16]. The values of 'analyzability and understandability' have been used. using SPSS, coefficients are calculated and model for testability quantification is thus developed as given below in equation (6).

Testability = -155.184+ 32.950× analyzability + 103.058× understandability ***Eq* (6)**

TABLE III

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.969[a]	.939	.899	4.60336
a. Predictors: (Constant), Understandability, Analyzability				

IX. Empirical Validation

In order to validate testability quantification model the data has been taken from [16].During experimentation, testability value of the projects has been computed using the testability model. These computed ratings are after that compared with the known rating specified by experts [2] with the help of Speraman's Coefficient of Correlation.

TABLE IV Computed Ranking, Actual Ranking and their Relation

Projects ↓	Testability Ranking		$\sum d^2$	r_s	$r_s >$
	Known Value	Known Rank			
p1	0.33	5	4	0.98	✓
p2	0.01	3	4	0.98	✓
p3	-0.23	2	25	0.85	✓
p4	1.95	10	16	0.90	✓
p5	-0.71	1	49	0.70	×
p6	0.98	6	25	0.85	✓
p7	1.95	9	49	0.70	×
p8	1.94	8	16	0.90	✓
p9	0.01	4	36	0.782	✓
p10	1.46	7	4	0.98	✓

The 'r_s' was computed using the technique given as

$$r_s = 1 - \frac{6\sum d^2}{n(n^2-1)} \qquad -1.0 \le r_s \le +1.0$$

'd' = difference between 'Calculated ranking' and 'Known ranking' of testability.

n = number of projects

The correlations are standard with high degree of confidence, i.e. up to 99% in eight cases out of ten. For that reason we can conclude without any loss of generality that testability quantification model are really significant and applicable in the context.

X. Conclusion

In this paper, software testability major factors are identified and their impact on testability quantification at design phase has been analyzed. 'Analyzability and Understandability', two of the major factors affecting testability have been taken into model development, and the statistical conclusions are validated for high level acceptability. Testability Quantification model has been validated theoretically as well as empirically using experimental test. The applied validation on the testability quantification model concludes that testability model is extremely consistent, satisfactory and reliable.

References

1. Poston, Robin, Jignya Patel, and Jasbir Dhaliwal. "A software testing assessment to manage project testability." (2012).
2. Abdullah, Dr, M. H. Khan, and Reena Srivastava. "Testability Measurement Model for Object Oriented Design (TMM^{OOD})." International Journal of Computer Science & Information Technology (IJCSIT) Vol. 7, No 1, February 2015, DOI: 10.5121/ijcsit.2015.7115.
3. ISO, "International standard ISO/IEC 9126. Information technology: Software product evaluation: Quality characteristics and guidelines for their use." 1991.
4. 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.
5. Lee, Ming-Chang. "Software Quality Factors and Software Quality Metrics to Enhance Software Quality Assurance." British Journal of Applied Science & Technology 4.21 (2014).
6. Abdullah, Dr, Reena Srivastava, and M. H. Khan. "Modifiability: A Key Factor To Testability", International Journal of Advanced Information Science and Technology, Vol. 26, No.26, Pages 62-71 June 2014.
7. Jianping, Fu, Liu Bin, and Lu Minyan. "Present and future of software testability analysis."International Conference on Computer Application and System Modeling (ICCASM 2010). Vol. 15.2010.
8. Abdullah, Dr, M. H. Khan, and Reena Srivastava. "Flexibility: A Key Factor To Testability", International Journal of Software Engineering & Applications (IJSEA), Vol.6, No.1, January 2015. DOI: 10.5121/ijsea.2015.6108.
9. Jungmayr S.: Testability during Design, Softwaretechnik-Trends, Proceedings of the GI Working Group Test, Analysis and Verification of Software, pp. 10-11, Potsdam. (2002).
10. Mahfuzul Huda, Dr.Y.D.S.Arya, and Dr.M. H. Khan. "Measuring Testability of Object Oriented Design: A Systematic Review." *International Journal of Scientific Engineering and Technology*, Vol. 3, Issue 10, pp: 1313-1319 Oct, 2014.
11. M. Bruntink and A. V. Deursen, "Predicting Class Testability Using Object-Oriented Metrics," Proceedings of IEEE International Workshop on Source Code Analysis and Manipulation, Chicago, 15-16 September 2004, pp. 136-145.
12. Abdullah, Dr, Reena Srivastava, and M. H. Khan. "Testability Estimation of Object Oriented Design: A Revisit". International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 8, pages 3086-3090, August 2013.
13. Dromey, R.G.: A Model for Software Product Quality. IEEE Transaction on Software Engineering 21(2), 146–162 (1995).
14. J. Gao, S. Ming-Chih, "A Component Testability Model for Verification and Measurement", In Proceedings of the 29th Annual International Computer Software and Applications Conference, pages 211–218.IEEE Computer Society (2005).
15. K.K. Aggarwal, Yogesh Singh. New Age International, Jan 1, 2005 - Software engineering.
16. M. Genero, J. Olivas, M. Piattini, and F. Romero, "A Controlled Experiment for Corroborating the Usefulness of Class Diagram Metrics at the Early Phases of Object-Oriented Developments," *Proc. of the ADIS 2001, Workshop on Decision Support in Software Engineering*, vol. 84. Spain, 2001.
17. Mahfuzul Huda, Dr.Y.D.S.Arya, and Dr. M. H. Khan. "Evaluating Effectiveness Factor of Object Oriented Design: A Testability Perspective." International Journal of Software Engineering & Applications (IJSEA), Vol.6, No.1, January 2015, DOI:10.5121/ijsea.2015.6104.
18. Binder, Robert V. "Design for testability in object-oriented systems." Communications of the ACM 37.9 (1994): 87-101.
19. Marinescu, Radu, and Daniel Ratiu. "Quantifying the quality of object-oriented design: The factor-strategy model." Reverse Engineering, 2004. Proceedings. 11th Working Conference on. IEEE, 2004.
20. S. Mouchawrab, L. C. Briand and Y. Labiche, "A Measurement Framework for Object-Oriented Software Testability," Information and Software Technology, Vol. 47, No. 1, 2005, pp. 979-997
21. 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.
22. Shahida Khatoon, Dr. R. Kumar. "Testibility Quantification of Object Oriented Design: A Revisit" , IJARCSSE, Vol. 4, Issue 10, Pages 524-529,October 2014, ISSN: 2277 128X.