Global Journals LaTeX JournalKaleidoscope™
Artificial Intelligence formulated this projection for compatibility purposes from the original article published at Global Journals. However, this technology is currently in beta. *Therefore, kindly ignore odd layouts, missed formulae, text, tables, or figures.*

Testability Assessment Model for Object Oriented Software based on Internal and External Quality Factors

Harsha Singhani[1] and Dr. Pushpa R. Suri[2]

[1] Kurukshetra University

Received: 16 December 2014 Accepted: 5 January 2015 Published: 15 January 2015

Abstract

Software testability is coming out to be most frequent talked about subject then the underrated and unpopular quality factor it used to be in past few years. The correct and timely assessment of testability can lead to improvisation of software testing process. Though many researchers and quality controllers have proved its importance, but still the research has not gained much momentum in emphasizing the need of making testability analysis necessary during all software development phases. In this paper we review and analyse the factors affecting testability estimation of object oriented software systems during design and analysis phase of development life cycle. These factors are then linked together in the form of new assessment model for object oriented software testability. The proposed model will be evaluated using analytical hierarchical process (AHP).

Index terms— software testability, testability factors, object oriented software testability assessment model.

1 Introduction

estability is one of the qualitative factors of software engineering which has been accepted in **??**cCall and Boehm software quality model, which build the foundation of ISO 9126 software quality model. Formally, Software testability has been defined and described in literature from different point of views IEEE [1] defines it 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" and ISO [2] has defined software testability as functionality or "attributes of software that bear on the effort needed to validate the software product".

In this paper we have proposed a testability evaluation model for assessment during design and analysis phase based on external quality factors and their relation with internal object oriented programming features which affect testability as shown earlier in our work [7].This paper is organized as follows: Section2 gives brief overview of software testability related work. Section3 gives the details of internal object oriented features needed for testability assessment followed by section 4 which gives the details of external quality factors linked and affected due to these features. Section 5 describes the proposed assessment model. It is followed by conclusion and future scope in section 6.

2 II. Software Testability Related Work

Software Testability actually acts as a software support characteristic for making it easier to test. As stated by Binder [8] and Freedman [9] a Testable Software is one that can be tested easily, systematically and externally at the user interface level without any adhoc measure. Whereas Voas [10] describe it as complimentary support to software testing by easing down the method of finding faults within the system by focussing more on areas that most likely to deliver these faults. Hence, over the years Testability has been diagnosed as one of the core quality indicators, which leads to improvisation of test process. The insight provided by testability at designing, coding and testing phase is very useful as this additional information helps in product quality and reliability improvisation **??**11][12]. All this has lead to a notion amongst practitioners that testability should be planned early in the design phase though not necessarily so. As seen by experts like Binder it involves factors like controllability and observability i.e. ability to control software input and state along with possibility to observe the output and state changes that occur in software. So, overall testable software has to be controllable and observable [8]. But Year 2015

3 ( )

4 C

The testability research actually is done from the prospect of reducing testing effort and testing cost which is more than 40% of total development cost of any software [3]. Still, the research in the field of testability has not been done in much detail. It mainly affects the efficiency of overall software development team from project managers, software designers to software testers. As they all need testability assessment in decision making, software designing, coding and testing [4].So keeping that in mind, we will take this study further. As discussed in our previous work about testability and testability metrics [5], [6], it has been found that testability research has taken a speed up in past few years only and much of the work has been done using various object oriented software metrics. over the years more such quality factors like understandability, traceability, complexity and testsupport capability have contributed to testability of a system [4].

Software testability measurement refers to the activities and methods that study, analyze, and measure software testability during a software product life cycle. Unlike software testing, the major objective of software testability measurement is to find out which software components are poor in quality, and where faults can hide from software testing. In the past, there were a number of research efforts addressing software testability measurement. Now these measurements can be applied at various phases during software development life cycle of a system. The studies mostly revolve around the measurement methods or factors affecting testability along with how to measure software testability at various phases like Design Phase [8], [12]- [18] and Coding Phase [19]- [22]. Lot of stress has been given upon usage of object oriented metrics for object oriented software testability evaluation during these researches. The metrics investigated related to object oriented software testability assessment mostly belong to static software metrics category. These metrics were mostly adapted from CK [23], MOOD [24], Brian [25], Henderson-Sellers [26] metric suite along with others [27]. Lot of empirical study has been done by researchers like Badri [28], Bruntink [29] and Singh [30] in showing the correlation of these metrics with unit testing effort. Few studies done by Baudry and Genero [31]- [34] have been focussed on UML diagram features from software testability improvisation prospect as found during review of these design diagrams. All this work has been explained in depth in our previous research work [4], [5].

We would take this study further keeping focus mainly on object oriented system as object oriented technology has become most widely accepted concept by software industry nowadays. But testability still is a taboo concept not used much amongst industry mainly due to lack of standardization, which may not be imposed for mandatory usage but just been looked upon for test support [35]. We would actually like to propose a model for testability evaluation based on key programming features and quality factors which in turn make testing easier or difficult within this software. We have followed the steps as mentioned below to formalize the model:

? Identification of internal design features for object oriented software testability assessment

? Identification of static metrics out of many popular metrics for each of these.

? Identification of external factors affecting software testability.

? Establishing link between theses external quality factors and internal features which are evaluated through selected object oriented metrics. ? Establishing link between testability and these identified external factors which indirectly link it to identified internal features. ? The Model is followed with evaluation using AHP technique.

5 III. Testability Factors Identification

Before proposing the testability assessment model we have to first identify the key object oriented programming features which affect the testability at internal level. As already known the object oriented programming is based on three core concepts-Inheritance, Encapsulation and Polymorphism. Where, Inheritance is a mechanism for code reuse and to allow independent extensions of the original software via public classes and interfaces. Whereas, Polymorphism mainly provides the ability to have several forms, and Encapsulation an after effect of information hiding is actually play significant role in data abstraction by hiding all important internal specification of an object and showing only external interface. Now, a programming without these characteristics is distinctly not objectoriented that would merely be programming with some abstract data types and structured coding [36]. But these are not the only factors directing the course of testing in object oriented software, along with them three more identified features namely coupling, cohesion and size complexity. All these features and their influence on testability has already been highlighted in our previous work [4], [5]. Hence these six identified object oriented programming core features would be necessarily required to assess testability for object oriented software at design level. All these internal quality characteristics -Encapsulation, Inheritance, Coupling, Cohesion, Polymorphism and Size & Complexity are as defined below in Table 1along with details of their specific relation on testability. The relation between these features and testability has been build based on thorough study of many publications [2], [20], [35], [38], [39] Cohesion is one of the measures of goodness or good quality in the software as a cohesive module is more understandable and less complex. Low cohesion is associated with traits in programming such as difficult to maintain, test, reuse, and even understand.

6 Size & Complexity

It's the measure of size of the system in terms attributes or methods included in the class and capture the complexity of the class.

Size & Complexity has a significant impact on understandability, and thus testability or maintainability of the system.

7 Polymorphism

Polymorphism allows the implementation of a given operation to be dependent on the object that "contains" the operation such that an operation can be implemented in different ways in different classes.

Polymorphism reduces complexity and improves reusability. More use of polymorphism leads more test case generation [29]. Now all the above mentioned key features can be measured by many object oriented metrics options available as discussed earlier in our previous article [6]. Most of these metrics are accepted by practitioners on 'heavy usages and popularity' and by academic experts on empirical (post development) validation. But to keep study simple from further evaluation perspective we have suggested the few basic but popular metrics amongst testability researchers. Out of all the popular metrics suites discussed in our previous work [41] few of these static metrics are as explained below in Table2 have been suggested for the evaluation of each of these feature and their effects on any object oriented software testability at design time.

As described in Table2 below for Encapsulation evaluation number of methods metrics (NOM) is being suggested by many researchers for the effect of information hiding on testability [16], [42]. So we kept it for encapsulation evaluation for our model too. Inheritance is evaluated either using Number of Children metrics (NOC) or Depth of Inheritance Tree (DIT) two of the most popular and efficient inheritance metrics [22], [36], [41], [42]. For Coupling we suggested coupling between objects (CBO) and for cohesion Li & Henry Cohesion between Methods metrics version (LCOM). These two were the most sought after and unparalleled metrics available for assessing coupling and cohesion effect on testability as per literature study and popularity amongst industry practitioners [10], [20], [22], [24], [37], [43].Though Size & Complexity can be easily measured by many metrics in this category such as number of classes (NOC) ,number of attributes (NOA), weighted method complexity (WMC) metrics but due to its significant role, popularity and association in number of test case indication pointed WMC is most appropriate [8], [28], [44]. Polymorphism is one of the underlying factors affecting testability but as quite stressed by early researchers like Binder and others [8], [25] as it results in testability reduction ,we suggest chose polymorphism factor metrics (POF/PF) one of the quick and reliable polymorphism evaluation method for testability assessment. Our proposed testability model is based on Dromey's software quality model [39] which has been a benchmark in use for various quality features as well as many testability models so far. So, as discussed above we have already highlighted all the internal design features from testability perspective as pointed by many researchers. These features directly or indirectly affect the quality factors which further make software may or may not more testable. The studies indicate encapsulation promotes efficiency and complexity. Inheritance has a significant influence on the efficiency, complexity, reusability and testability or maintainability. While low coupling is considered good for understandability, complexity, reusability and testability or maintainability, whereas higher measures of coupling are viewed to adversely influence these quality attributes. Cohesion is viewed to have a significant effect on a design's understandability and reusability.

8 Global Journal of C omp uter S cience and T echnology

Size & Complexity has a significant impact on understandability, and testability or maintainability. Polymorphism reduces complexity and improves reusability. Out of six identified features four features have been proposed in MTMOOD testability model [16], which does not cover the polymorphism and size & complexity feature, which have also been found as essential internal features by many researchers in testability study [15], [22], [36], [37]. These six object oriented features play a very significant role in testability improvisation directly or indirectly through other quality factors.

All the above mentioned studies lead to mainly six identified external quality factors to assess testability for object oriented software. These factors are -Controllability, Observability, Complexity, Understandability, Traceability and Built-in-Test. Most of these factors were pointed in Binder's [8] research work on testability. Many other researchers established these factors relation too with testability as mentioned below in table 3.We have identified these factors keeping in mind significant role in testability as found out in our previous research work and surveys e have identified These factors get directly or indirectly affected by all of the above mentioned internal features and further complicate or reduce the task of testing hence reducing or increasing overall testability of the software. Controllability is an important index of testability as it makes testing easier [9], [47]- [49].

9 Observability

Software observability indicates how easy to observe a program in terms of its operational behaviours, input parameters, and outputs. In the process of testing, there is a need to observe the internal details of software execution, to ascertain correctness of Observable software makes it feasible for the tester to observe the internal behaviour of the software, to the required degree of details, Hence observability increases testability in the system [9], [47], [49].

12 Global Journal of C omp uter S cience and T echnology Volume XV Issue V Version I Year ( )C

processing and to diagnose errors discovered during this process possibility to observe the output and state changes that occur in software.

10 Complexity

It is basically described as the difficulty to maintain, change, understand and test software.

High Complexity of the system is actually an indicator of decreased system testability [43], [42], [50], [51]. Understandability It is the degree to which the component under test is documented or self-explaining.

An understandable system is easily testable and [14], [52]- [54]. Traceability It is the degree to which the component under test is traceable in other words the requirements and design of a given software component match.

A non-traceable software system cannot be effectively tested, since relations between required, intended and current behaviours of the system cannot easily be identified [8], [44]. Built In Test(BIT)

Built in testing involves adding extra functionality within system components that allow extra control or observation of the state of these components.

BIT actually provides extra test capability within the code for separation of test and application functionality which makes software more testable by better controllability and improved observability [8], [19], [55], [56].

Now after listing all the internal object oriented programming features which directly affect testability and all external quality factors which are also indicators of testable software, we have to identify the link between the two. As found on the basis of above literature survey the influence of all internal features over external quality features is briefly explained below in Table 4 below: â??" Low I -High U â??" Low Cp- High U ? High Ch- High U â??" Big size - Low U - Traceability (T) â??" High E - Low T - â??" High Cp- Less T - â??" Low Size - More T - Built In test (BIT) ? High E -More BIT - ? High Cp- More BIT â??" High Ch- Less BIT - -

The table actually elaborates the contribution of each of these internal programming features towards the six major quality factors which are directly linked to testability. Hence we may say that Testability requires Low Coupling, Adequate Complexity, Good Understandability, High Traceability, Good observability, Adequate control and more Built in test. In spite of having lot of measurement techniques for testability evaluation using some or the factor or few of the above mentioned metrics, testability has not yet been found to be evaluated from these factor perspectives. The study still does not show an elaborative impact of all of them together for testability improvisation or test effort reduction which is what motivated us for proposing this new model.

So, the proposed testability assessment model with respect to internal design features using static metrics is based on six above mentioned object oriented features from testability perspective as pointed in Binders research too [8]. The proposed model is as follows:

11 Global

12 Conclusion & Future Scope

In this paper an evaluation model for testability assessment during design and analysis phase based on external factors and their relation with internal object oriented programming features has been proposed. These factors directly or indirectly affect testability and can be used for software testability measurement. On the basis of detailed study we may say that Testability requires Low Coupling, Adequate Complexity, Good Understandability, High Traceability, Good observability, Adequate control and more Built in test. [1] [2]

1

Figure 1: CFigure 1 :

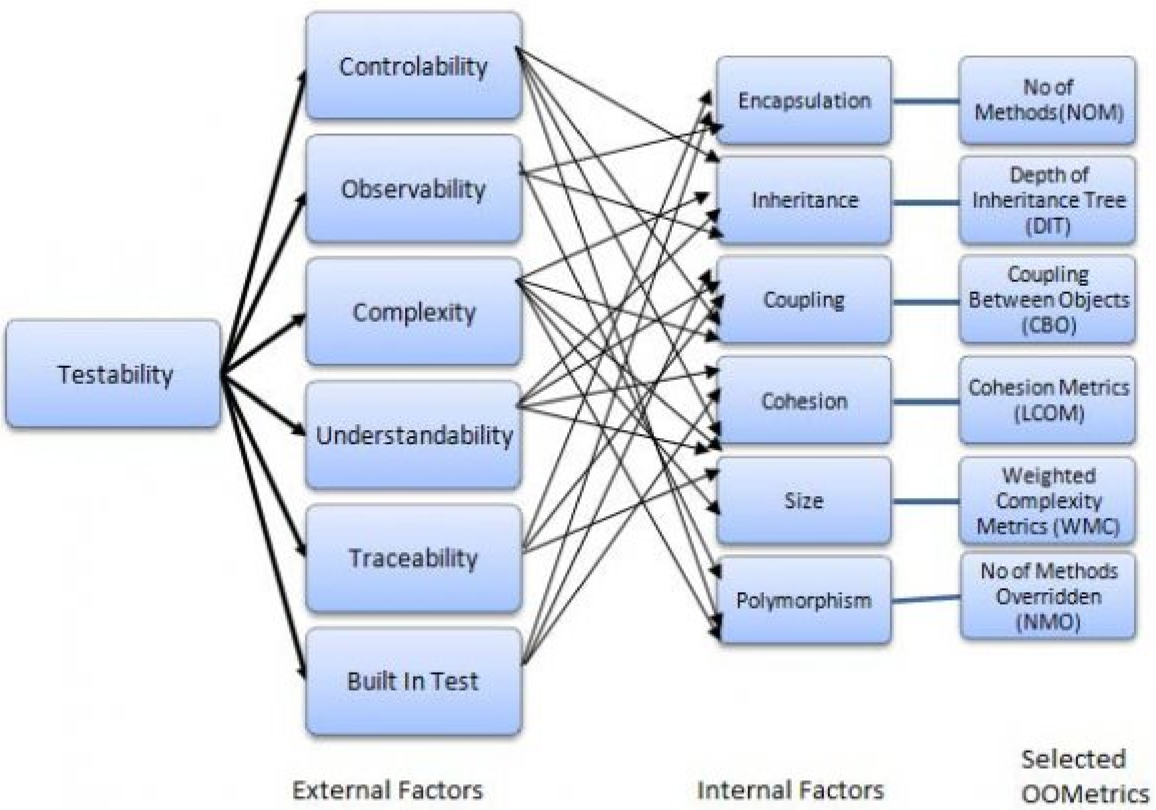

Figure 2:

1

10

[Note: C]

Figure 3: Table 1 :

2

Year 2015
Volume XV Issue V Version I
( )

[Note: C]

Figure 4: Table 2 :

3

External Factors Affecting Testability	Definition	Significant Testability Relation in Literature
Controllability	It is the ability to control software input and state. During software testing, some conditions like disk full, network link failure etc. are difficult to test. Controllable software makes it possible to initialize the software to desired states, prior to the execution of various tests.	

Figure 5: Table 3 :

4

	Affecting Testability					
	Encapsulation (E)	Inheritance (I)	Coupling (Cp)	Cohesion (Ch)	Size (S)	Polymorphism (P)
Controllability (Ct)	â??”		â??”	?		â??”
	High E-Low Ct	-	High Cp - Low Ct	High Ch- High Ct	-	High P-Low Ct
Observability (O)	â??”	?				â??”
	High E - Low O	High I -High O	-	-	-	High P-Low O
Complexity (Cx)		â??”	?	â??” High	?	â??”
	-	Low I -High Cx	High Cp- More Cx	Ch - Reduce Cx	Big S- More Cx	High P - Reduce Cx
Understandability (U)	-					

Figure 6: Table 4 :

[Radatz et al. ()] , J Radatz , A Geraci , F Katki . *IEEE Standard Glossary of Software Engineering Terminology* 1990. 610 p. . (IEEE Std)

[Softw and Eng ()] , Softw , Eng . 1994. 20 p. .

[Henderson and Sellers ()] , B Henderson , Sellers . 1996. New Jersey: Prentice Hall. (Object-Oriented Metric)

[Booch et al. ()] , G Booch , R A Maksimchuk , M W Engle , B J Young , J Conallen , K A Houston . *Object-Oriented Analysis and Design with Applications* 2007. Addison Wesley. 1 (11) .

[Int and Conf ()] , Int , Conf . *Comput. Appl. Syst. Model. Proc* 2010. 15. (Iccasm)

[Mouchawrab et al. (2005)] 'A measurement framework for object-oriented software testability'. S Mouchawrab , L C Briand , Y Labiche . *Inf. Softw. Technol* April. 2005. 47 p. .

[Nazir et al. ()] 'A Metrics Based Model for Understandability Quantification'. M Nazir , R A Khan , K Mustafa . *J. Comput* 2010. 2 (4) p. .

[Chidamber and Kemerer] 'A Metrics Suite for Object Oriented Design'. S R Chidamber , C F Kemerer . *IEEE Trans*

[Dromey ()] 'A Model for Software Product Quality'. R G Dromey . *IEEE Transactions on Software Engineering* 1995. 21 p. .

[Genero et al. ()] 'A survey of metrics for UML class diagrams'. M Genero , M Piattini , C Calero . *J. Object Technol* 2005. 4 (9) p. .

[Badri et al. ()] 'An empirical analysis of a testability model for object-oriented programs'. M Badri , A Kout , F Toure . *ACM SIGSOFT Softw. Eng. Notes* 2011. 36 (4) p. 1.

[Badri et al. ()] 'An empirical analysis of lack of cohesion metrics for predicting testability of classes'. L Badri , M Badri , F Toure . *Int. J. Softw. Eng. its Appl* 2011. 5 (2) p. .

[Voas et al. ()] 'An Empirical Comparison of a Dynamic Software Testability Metric to Static Cyclomatic Complexity'. J M Voas , J M Voas , K W Miller , K W Miller , J E Payne , J E Payne . *Proc. 2nd Int'l. Conf. Softw. Qual. Manag*, (2nd Int'l. Conf. Softw. Qual. Manag) 1994. p. .

[Bruntink and Vandeursen ()] 'An empirical study into class testability'. M Bruntink , A Vandeursen . *J. Syst. Softw* 2006. 79 p. .

[Genero et al.] *An Empirical Study to Validate Metrics for Class Diagrams*, M Genero , M Piattini , C Calero .

[Nazir and Mustafa ()] 'An Empirical Validation of Testability Estimation Model'. M Nazir , K Mustafa . *Int. J. Adv. Res. Comput. Sci. Softw. Eng* 2013. 3 (9) p. .

[Khalid et al. ()] 'Analysis of object oriented complexity and testability using object oriented design metrics'. S Khalid , S Zehra , F Arif . *Proceedings of the 2010 National Software Engineering Conference on -NSEC '10*, (the 2010 National Software Engineering Conference on -NSEC '10) 2010. p. .

[Goel et al. ()] 'COTT -A Testability Framework for Object-Oriented Software Testing'. A Goel , S C Gupta , S K Wasan . *Int. Jounal Comput. Sci* 2008. 3 (1) p. .

[Patidar et al. ()] 'Coupling and Cohesion Measures in Object Oriented Programming'. M Patidar , R Gupta , G Chandel . *Int. J. Adv. Res. Comput. Sci. Softw. Eng* 2013. 3 (3) p. .

[Saaty ()] 'Decision making with the analytic hierarchy process'. T L Saaty . *Int. J. Serv. Sci* 2008. 1 (1) p. 83.

[Mulo ()] *Design for testability in software systems*, E Mulo . 2007.

[Binder ()] 'Design For Testabity in Object-Oriented Systems'. R Binder . *Commun. ACM* 1994. 37 p. .

[Fernando ()] 'Design Metrics for OO software system'. A Fernando . *ECOOP'95, Quant. Methods Work*, 1995.

[Esposito ()] *Design Your Classes For Testbility*, D Esposito . 2008.

[Badri (2012)] 'Empirical Analysis of Object-Oriented Design Metrics for Predicting Unit Testing Effort of Classes'. M Badri . *J. Softw. Eng. Appl* July. 2012. 05 p. .

[Sheppard and Kaufman ()] 'Formal specification of testability metrics in IEEE P1522'. J W Sheppard , M Kaufman . *IEEE Autotestcon Proceedings. IEEE Syst. Readiness Technol. Conf. (Cat. No.01CH37237)*, 2001. 2001. p. .

[Bach ()] *Heuristics of Software Testability*, J Bach . 2003. p. 2003.

[Voas and Miller ()] 'Improving the software development process using testability research'. J M Voas , K W Miller . *Softw. Reliab. Eng* 1992. ?, 1992.

[Baudry et al. ()] 'Improving the testability of UML class diagrams'. B Baudry , Y Le Traon , G Sunye . *First Int. Work. onTestability Assessment* 2004. IWoTA 2004. 2004. (Proceedings.)

[Kansomkeat et al. ()] 'INCREASING CLASS-COMPONENT TESTABILITY'. S Kansomkeat , J Offutt , W Rivepiboon . *Proceedings of 23rd IASTED International Multi-Conference*, (23rd IASTED International Multi-Conference) 2005. p. .

[Jeon ()] *Increasing the Testability of Object-Oriented Frameworks with Built-in Tests*, T Jeon . 2002. p. . (Building)

[Briand et al. ()] ‘Investigating quality factors in objectoriented designs: an industrial case study’. L C Briand , J Wust , S V Ikonomovski , H Lounis . *Proc. 1999 Int. Conf. Softw. Eng. (IEEE Cat. No.99CB37002)*, (1999 Int. Conf. Softw. Eng. (IEEE Cat. No.99CB37002)) 1999.

[ISO/IEC 9126: Software Engineering Product Quality ()] *ISO/IEC 9126: Software Engineering Product Quality*, 2002.

[Mathur ()] A P Mathur . *Foundations of Software Testing, Second. Pearson*, 2013.

[Baudry and Le Traon ()] ‘Measuring design testability of a UML class diagram’. B Baudry , Y Le Traon . *Inf. Softw. Technol* 2005. 47 (13) p. .

[Mayer and Hall ()] ‘Measuring OO systems: a critical analysis of the MOOD metrics’. T Mayer , T Hall . *Proc. Technol. Object-Oriented Lang. Syst. TOOLS* 1999. 29.

[Khan and Mustafa ()] ‘Metric based testability model for object oriented design (MTMOOD)’. R A Khan , K Mustafa . *ACM SIGSOFT Softw. Eng. Notes* 2009. 34 (2) p. 1.

[Nguyen et al. ()] T B Nguyen , M Delaunay , C Robach . *Testability Analysis of Data-Flow Software*, 2005. 116 p. .

[Khan and Khan ()] ‘Object Oriented Design Complexity Quantification Model’. S A Khan , R A Khan . *Procedia Technol* 2012. 4 p. .

[Suri and Singhani ()] ‘Object Oriented Software Testability (OOSTe) Metrics Analysis’. P R Suri , H Singhani . *Int. J. Comput. Appl. Technol. Res* 2015. 4 (5) p. .

[Suri and Singhani ()] ‘Object Oriented Software Testability Survey at Designing and Implementation Phase’. P R Suri , H Singhani . *Int. J. Sci. Res* 2015. 4 (4) p. .

[Singhani and Suri ()] ‘Object Oriented SoftwareTestability (OOSTe) Metrics Assessment Framework’. H Singhani , P R Suri . *Int. J. Adv. Res. Comput. Sci. Softw. Eng* 2015. 5 (4) p. .

[Wang et al. ()] ‘On testable object-oriented programming’. Y Wang , G King , I Court , M Ross , G Staples . *ACM SIGSOFT Softw. Eng. Notes* 1997. 22 (4) p. .

[Singh and Saha ()] ‘Predicting Testability of Eclipse: Case Study’. Y Singh , A Saha . *J. Softw. Eng* 2010. 4 (2) p. .

[Fu et al.] *Present and future of software testability analysis*, J Fu , B Liu , M Lu . ICCASM 2010 -2010.

[Vincent and King ()] *Principles of Built-In-Test for Run-Time-Testability in Component-Based Software Systems*, J Vincent , G King . 2002. p. .

[Nazir and Khan ()] ‘Software Design Testability Factors: A New Perspective’. M Nazir , R A Khan . *Proceddings of Naional Third Conference INDIACOM*, (eddings of Naional Third Conference INDIACOM) 2009. 2009. p. .

[Rosenberg and Hyatt ()] *Software quality metrics for object-oriented environments*, L Rosenberg , L Hyatt . 1997.

[Voas and Miller ()] ‘Software Testability : The New Verification’. J M Voas , K W Miller . *IEEE Softw* 1995. 12 (3) p. .

[Bach ()] *Test Plan Evaluation Model*, J Bach . 1999. p. .

[Baudry et al. ()] ‘Testability analysis of a UML class diagram’. B Baudry , Y Le Traon , G Sunye . *Proc. Eighth IEEE Symp. Softw. Metrics*, (Eighth IEEE Symp. Softw. Metrics) 2002.

[Jungmayr and Pettichord ()] *Testability during Design*, S Jungmayr , ; B Pettichord . 2002. 2002. p. . (Design for Testability. Pettichord.com)

[Nazir et al. ()] ‘Testability Estimation Framework’. M Nazir , R A Khan , K Mustafa . *Int. J. Comput. Appl* 2010. 2 (5) p. .

[Bruntink ()] *Testability of Object-Oriented Systems : a Metrics-based Approach*, M Bruntink . 2003. Universiy Van Amsterdam

[Freedman ()] ‘Testability of software components -Rewritten’. R S Freedman . *IEEE Trans. Softw. Eng* 1991. 17 (6) p. .

[Harman et al. ()] ‘Testability Transformation: Program Transformation to Improve Testability’. M Harman , A Baresel , D Binkley , R Hierons . *Formal Method and Testing*, 2011. p. .

[Payne et al. ()] *The above proposed model requires to be evaluated using some technique which helps in validating these criteria’s, sub-criteria’s and their significant quantifiable role in testability assessment. We may use one of the formal Multi criteria decision making (MCDM) technique proposed by Satty*, J E Payne , R T Alexander , C D Hutchinson . 1997. 7 p. . (Design-for-Testability for Object-Oriented Software. 57] known as Analytic Hierarchy Process (AHP)

[Baudry et al. ()] ‘Towards a ’ Safe ’ Use of Design Patterns to Improve OO Software Testability’. B Baudry , Y Le Traon , G Sunye , JM . *ISSRE 2001. Proceedings. 12th Int. Symp*, 2001. 2001. p. .