DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4245

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

REQUIREMENT UNDERSTANDABILITY QUANTIFICATION MODEL OF OBJECT ORIENTED SOFTWARE

Mohammad Zunnun Khan
Departement of Computer Science & Engineering,
Integral University,
Lucknow, India

M Akheela Khanam
Department of Computer Science & Engineering,
Integral University,
Lucknow, India

M H Khan
Department of Computer Science & Engineering,
IET, Sitapur Road,
Lucknow, India

Abstract: Understandability has significant role in development of quality software; it incredibly impacts cost, quality and unwavering quality at the time software development (especially at early stages of development). Wrong interpretation prompts ambiguities, misconception and thus the misinterpretations of further development process and the related records, which frequently results to defective development. Notwithstanding the way that understandability is essential and very critical viewpoint for software development process. Understandability is considered as basic building block for delivering high quality and reliable software. It greatly influences cost, quality and reliability at the time of software evolution. In this paper, author highlights the importance of understandability early at requirement phase in general and as a factor of software testability. The paper quickly portrays the proposed model for understandability quantification of object oriented software [RUM^{OOS}] by establishing multiple linear regressions. Finally the proposed model has been validated using experimental tryout.

Keywords: Understandability, UML, Software Testability, Requirement Quality, Object Oriented Software

1. INTRODUCTION

In the recent years, software developers have put special attention to guarantee the quality characteristics of object oriented systems [1] ,[27] . Quality has turned out to be more essential with our expanding reliance on software. In the most recent decades the interest for quality in a software product has been progressively underscored [29] . Software industry has been conveying exponential change in cost, execution, yet the issues with software are not declining. According to the one of the IBM report, around 30% of the undertakings get drop before they are finished, 52% over-run their cost gauges by a normal of 189%, and for every 100 projects, there are 94 restarts [31] [2] . A key issue of software industry is its absence of capacity to create bug free software. On the off chance that software engineers are asked to authoritatively express that the created software is sans bug, no product would have ever been discharged. Target of software engineering is to make great final software product in time and within proposed budget. On the off chance that an item is meeting its necessities, we may state it is an unrivalled quality product. The entire thing is measured concerning requirements and in the event that it matches, product is a quality product [3] [27] .

Software has turned out to be a key to progression in every aspect of human attempt. The capacity of programming just is no longer tasteful to build extensive projects. There are real issues in the cost, opportunity, support and nature of numerous software products. Software engineer has the objective of taking care of these issues by creating great quality, testable, maintainable software, on time, inside spending plan [4] ,[27] . As per software engineering standards, if the procedure for development of any software product is correct, the possibility of accomplishment of the software product undertakings is enormously increased. To accomplish this target, study has to focus in a trained way around both the quality of the product and on the procedure used to build up the product. Nonetheless, because of increment in cost of testing and upkeep of software, goal is presently changing to convey quality software [5] ,[31] . Software testing is an essential and fundamental movement of development life cycle for delivering great quality software.

Testing is critical and testing errand. The time spent and exertion required for testing of software is extremely huge and expends around 40% to half of the aggregate cost for the complete development life cycle. The most imperative issue amid testing is that before revising a program (error), the developer should first follow and comprehends it and it is conceivable with the assistance of its understandability [5] [6] . It is vital that cost effective testing procedure must be connected amid development life cycle and maintenance. The essential component adding to the development of these practical strategies is the testability of software [27] . Software testability is characterized as a measure of the exertion required to palatably test the program as per some very much characterized testing criteria[7] [31] . To a huge viewpoint, testing relies upon how troublesome the mistake is to follow. Software testability and fault or error traceability are two most essential ideas: the more troublesome a blunder is to follow, the more troublesome it is with a specific end goal to be settled [29] . The more

troublesome it is to remedy, the higher its testability hazard is. The general exertion spent on testing not just relies upon human components; prepare issues, test methods, and test tools, additionally on attributes of the software development curios [8] [27] . How much a product ancient rarity encourages test assignments in a given test context is called testability.[13]

On the off chance that we need to enhance testability we need to follow those parts of a development that need testability [15] . In perspective of the reality, obviously adaptability holds a critical place as a component of testability and traceability is an essential paradigm of adaptability [8] [9] . The analyzer can utilize testability data to decide on what code to centre amid testing [9] . Testability has been recognized as one of the significant issues in the field of programming building for delivering excellent programming. It gives experiences that are observed to be particularly significant for the length of programming configuration, coding, testing and quality confirmation [10] .

2. SOFTWARE TESTABILITY

Testability is a standout amongst the most vital quality pointers; its estimation prompts to the possibilities of encouraging and enhancing a test procedure. The knowledge gave by software testability is important amid designing, coding, testing, and quality assurance[17] . The qualities of testable software like sufficient multifaceted nature, low coupling and great partition of concerns make it less demanding for reviewer to comprehend the product antiquities under survey [5] . Testability comes about because of good Software Engineering rehearse and a viable software process[10] [11] In spite of the fact that, testability is most clearly significant amid testing, however focusing on testability right on time in the development process, testing proficiency and adequacy may possibly be progressed[17] [18] . Testability can be seen as the property or potentially trademark that measures the simplicity of testing a bit of code or usefulness, and an arrangement included programming so test plans and scripts can be executed systematically [12] . Testability investigation can include data that is helpful both for surveying the general quality and for finding software bugs [13] .Consequently; it gives a trade-off investigation instrument to creators to help them in choosing whether they will pay the punishment for testability at the cost of different advantages.

3. OBJECT ORIENTED CHARACTERISTICS

In today's development environments, Object oriented analysis & very next stages are the well known ideas. They are frequently proclaimed as the silver shot for taking care of software problem while as a general rule there is no silver slug. In any case, it has demonstrated its esteem for system that must be maintained & modified [16] [22] . Requirement choices are made for various reasons, so the wording is interpreted in an unexpected way [19] . For example, in requirement modeling we discuss decomposition, abstraction as well as separation of concerns –all of which were initially design techniques for making elegant, modular designs[21] [23] . We decompose requirements specification along separate concerns to simplify the result based model and make it easier to read & understand [25] . Interestingly, we decompose a design outline to enhance the framework's quality attributes such as modularity, maintainability, performance & time bound delivery [26] [27] [28] .The requirements name and oblige those attributes, but decomposition has no role in this specification aspect. Thus, although we use the terms decomposition and modularity in specification as well as design, the decomposition decisions we make at each stage has different aspect because they have different goals[15] .Early in the requirement stage, it is less demanding to build an applied model of the issue that distinguishes what objects or entities are included, what they resemble (by characterizing their attributes), and how they identify with each other[16] . Such a model assigns names for the fundamental components of the issue [29] [30] . These components are then reused in different depictions of the requirement.

A. Establishing Relationship. To build up a relevant effect relationship between Object Oriented (OO) Software attributes and testability factors, the impact of OO characteristics on every factor of testability was inspected by a few scientists [31] . The vast majority of the studies cantered their endeavor to inspect the effect of OO characteristics and have effectively settled established with quality factors [30] . Be that as it may, we inspected and evaluated their effect on the specific part of study i.e. testability and by associatively and congruence viewpoint, finished up on recognizing testability factors influenced by Object characteristics [31] [32] . It was watched that each of these attributes, either have positive or negative effect on the factors that influence testability of OO development. After a thorough survey of accessible writing on the theme, the connection between OO development attributes and testability elements (as delineated in Figure1) has been set up[31] [5] [17] [18] [20] . In light of the relationship demonstrated as follows, a model has been created in area (condition 2) for assessing Understandability. Promote the relative noteworthiness of individual plan properties that impact software testability is weighted relatively [31] [32] . The idea of various straight relapses has been utilized to get the coefficients that build up a relationship between ward factors and numerous independent variables.

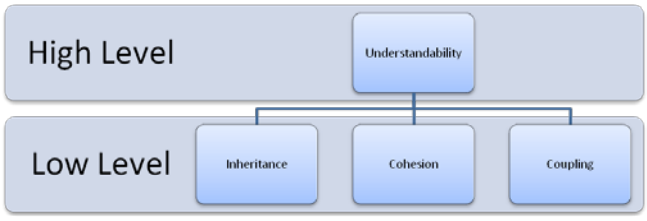

FIGURE 1. Requirement Understandability Quantification Model

The descriptive quality model (Jagdeesh Bansiya's hierarchal quality model [29]) has been thought-about as a basis to develop the Requirement Understandability Model for Object Oriented Software as shown in Figure 1. The proposed model establishes discourse impact relationship between Understandability and object oriented characteristics of software and their related metrics. The values of these metrics can be easily identified with the help

of UML diagram. This model used the low level object oriented metrics namely Afferent Coupling (CA), Measure of functional Abstraction (MFA), Direct Class Metric (DAM)[10] [27] [29] , to describe a range of measurement for software and defined in terms of design characteristic and also helpful for quantitative assessment of degree to which system, component or process hold a given attribute. Using Statistical Analysis software named as 'SPSS' values of all its independent variables (metrics), regression intercept and coefficient of the respective independent variables are calculated. On the basis of the multiple linear regression equation concepts, Requirement Understandability model has been developed that is given in equation (2). Factor of a class depend upon one or more number of object oriented software metrics, quality factor may be fixed by using model 'Requirement Understandability Quantification Model of Object oriented Software- RUM^{OOS}.'

$$Understandability = \alpha_0 \pm \beta_1 * Inheritance \pm \beta_2 * Cohesion \pm \beta_3 * Coupling \quad (1)$$

Where this equation has

- β_1, β_2 and β_3 are the coefficients of respective independent variables 'Requirement _ Inheritance, Requirement _Cohesion and Requirement _Coupling' related to understandability.

- α_0 is the intercept.

The data used for establishing Understandability model is taken from [32] that have been collected through large commercial object oriented systems as shown in Table1.

TABLE **1: Understandability Computation Table**

Project	**CA**	**MFA**	**CAM**	**Standard Understandability**
P_1	1	0	0.346939	1.582283
P_2	0	0	0.6	1.53945
P_3	0	0	0.5	1.5054
P_4	0	0	0.5	1.5054
P_5	1	0	0.2125	1.536506

$$Understandability = 1.50 + 0.0070 * ca + 0.35 * mfa + 0.121 * cam \quad (2)$$

4. ANALYSIS OF UNDERSTANDABILTY QUANITIFCATION MODEL

4.1Statistical Significance of Model

TABLE 2.Statistical Significance of Requirement Understandability Model- RUM^{OOS}

Descriptive Statistics			
	Mean	Std. Deviation	N
CAL	1.6308	.13688	10
CA	1.000000	.9428090	10
MFA	.2289	.36859	10
CAM	.3359	.19362	10

The descriptive table is very important for further research work. It gives the valuable record of descriptive statistics that are mean, standard deviation and number of samples selected for model validation.

TABLE 3. Correlation between Independent Variables

Correlations					
		CAL	CA	MFA	CAM
Pearson Correlation	CAL	1.000	.649	.985	.293
	CA	.649	1.000	.729	-.401
	MFA	.985	.729	1.000	.134
	CAM	.293	-.401	.134	1.000

Table 4. Model Summary for Understandability Model

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.999[a]	.998	.997	.00733
a. Predictors: (Constant), CAM, MFA, CA				

Summary table 4 for Understandability Quantification Model proves that all the four selected metrics are statistically significant at confidence level of 95%.

5. EMPIRICAL VALIDATION OF UNDERSTANDABILITY MODEL.

This section of work proves that how significant proposed study, where metrics and model are able to estimate the understandability quality index of object oriented at requirement time. The empirical validation is important phase of research to evaluate the proposed understandability quality model for high level acceptability and appropriate execution. Empirical validation is the fine approach and best practice for claiming the model acceptance [19] . To justify claiming approach for acceptance of model, an experimental validation of the proposed understandability quantification model at requirement time has been carried out using samples.

A. Data Set for Ten Projects. This paper describes an analysis that was conducted on collected repository with 92 versions of 38 proprietary, open-source and academic projects[32] [6] . In view of this fact, an experimental validation of the proposed model for Understandability evaluation has been carried out using sample tryouts. In order to validate proposed Understandability quantification

model, the value of metrics are available by using [6] [32] data set for following 10 projects in table 5.

TABLE 5. Known and Calculated Understandability Index Values and Ranking for 10 Projects

Project	**CA**	**MFA**	**DAM**	**STD_UNDERSTANDABILTY**	**CAL_ UNDERSTANDABILTY**
P_1	2	0.77	0.3	1.82	1.73
P_2	1	0	0.14	1.52	1.51
P_3	1	0	0.15	1.53	1.52
P_4	1	0	0	1.51	1.46
P_5	1	0.76	0.5	1.84	1.67
P_6	0	0	0.5	1.58	1.51
P_7	3	0.76	0.32	1.82	1.87
P_8	1	0	0.35	1.56	1.58
P_9	0	0	0.6	1.57	1.54
P_{10}	0	0	0.5	1.56	1.51

It is compulsory to test the validity of proposed model for acceptance. A 2 sample t test applies for check the significance between standard Understandability and calculated Understandability. 2t-test is handy hypothesis tests in statistics when compare means.

TABLE 6. 2 t- tests between Std_ Understandability & Cal_ Understandability

Paired Samples Statistics					
		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	CAL	1.6308	10	.13688	.04328
	KNO	1.5891	10	.12777	.04040

Null hypothesis (H_0): There is no significant difference between STD_UNDERSTANDABILTY and CAL_ UNDERSTANDABILTY**; H_0: μ1-μ2 = 0**

Alternate hypothesis (H_A): There is significant difference between STD_UNDERSTANDABILTY and CAL_ UNDERSTANDABILTY; **H_A: μ1-μ2 ≠ 0**

In the above hypothesis μ1 and μ2 are treated as sample means of population. Mean value and Standard Deviation value have been calculated for specified two samples and represented in table 6. Correlation comes out to be 0.901, that shows the standard Understandability and calculated Understandability is highly correlated. The hypothesis is tested with zero level of significance and 95% confidence level. The p value is 0.055. Therefore alternate hypothesis directly discards and the null hypothesis is accepted. The developed equation used for Understandability estimation is accepted.

6. CONCLUSION

The paper highlighted the importance of software Understandability and an approach is presented for assessing Understandability of requirements based on the collection of requirement quality measures of object oriented software at low level. Understandability is obviously relevant to the context of software testability and plays a highly significant role for delivering quality software. Subsequently, proposed an Understandability equation to obtained multivariate linear model have been measured for the Understandability of requirement. It has been shown that model-RUM^{OOS} is able to quantify the Understandability of the software requirement. Hence, therefore, the model has been validated theoretically as well as empirically using experimental try-out. However, the model is validated on a small data set and it is to be done further on live industrial projects for better acceptability and utility.

Acknowledgment. I would like to express my sincere gratitude to Integral University, Lucknow that provides me such a wonderful opportunity(with Communication no IU/R&D/2017-MCN0001)and to my supervisor Associate Professor Dr. M Akheela Khanam & Co-supervisor Prof. (Dr.) M. H. Khan for the continuous support of my PhD study and research, motivation, enthusiasm. Their guidance helped me in all the time of research. Last but not the least; I would like to thank my parent for their patience, understanding and support that drive me to complete my study.

REFERENCES

[1] I. Sommerville, Software Engineering, 9th edition. Boston, Massachusetts: Addison- Wesley, 2010, pp. 27–74.

[2] D. Gallin, Software Quality Assurance from Theory to Implementation. Edinburgh Gate: Pearson Education, 2004.

[3] J. Ramos, Ricardo; Piveta, Eduardo K; Castro, Jaelson; Moreira, Ana; Guerreiro, Pedro; Pimenta, Marcelo S; Price, R. Tom; Araujo, "Improving the Quality of Requirements with Refactoring," in Simposio Brasileiro de Qualidade de Software, 2009.

[4] D. Firesmith, "Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them," J. OBJECT Technol., Vol 6, No. 1, pp. 17–33, 2007.

[5] Mohammad Zunnun Khan, M.Akheela Khanam, M. H. K. (2016). Software Testability in Requirement Phase: A Review. International Journal of Advanced Research in Computer and Communication Engineering, 5(4), 1031-1035. DOI 10.17148/IJARCCE.2016.54252

[6] Genero, M., Piatini, M., & Manso, E. (2004, August). Finding" early" indicators of UML class diagrams understandability and modifiability. In Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on (pp. 207-216). IEEE.

[7] R. V. Binder, "Design for Testability in Object-oriented Systems," Commun ACM, vol. 37, no. 9, pp. 87–101, Sep. 1994.

[8] S. Hesari, R. Behjati, and T. Yue, "Towards a systematic requirement-based test generation framework: Industrial challenges and needs," in Requirements Engineering Conference (RE), 2013 21st IEEE International, 2013, pp. 261–266.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," SIGKDD Explor Newsl, vol. 11, no. 1, pp. 10–18,

Nov. 2009.

[10] Davis, A.; Overmyer, S.; Jordan, K.; Caruso, J.; Dandashi, F.; Dinh, A.; Kincaid, G.; Ledeboer, G.; Reynolds, P.; Sitaram, P.; Ta, A.; Theofanos, M., "Identifying and measuring quality in a software requirements specification," Software Metrics Symposium, 1993. Proceedings, First International, vol., no., pp.141,152, 21-22 May 1993

[11] Boehm, Barry, Guidelines for Verifying and Validating Software Requirements and Design Specifications, vol. Euro IFIP 79. North Holland, 1979.

[12] Cleland-Huang, Jane, Adam Czauderna, Alex Dekhtyar, Olly Gotel, Jane Huffman Hayes, Ed Keenan, Greg Leach et al. "Grand challenges, benchmarks, and TraceLab: developing infrastructure for the software traceability research community." In Proceedings of the 6th international workshop on traceability in emerging forms of software engineering, pp. 17-23. ACM, 2011.

[13] Cleland-Huang, Jane, Raffaella Settimi, Xuchang Zou, and Peter Solc. "Automated classification of non-functional requirements." Requirements Engineering 12, no. 2 (2007): 103-120.

[14] Sultanov, Hakim, and Jane Huffman Hayes, "Application of reinforcement learning to requirements engineering: requirements tracing." In Requirements Engineering Conference (RE), 2013 21st IEEE International, pp. 52-61, IEEE, 2013.

[15] Shahid Iqbal and Naeem Ahmed Khan M. Yet another Set of Requirement Metrics for Software Projects. International Journal of Software Engineering and Its Applications. 2012; 6.1:19-28.

[16] Bokhari Mohammad Ubaidullah and Shams Tabrez Ubaidullah Siddiqui. Metrics for Requirements Engineering and Automated Requirements Tools. Proceedings of the 5th National Conference, INDIACom-2011.

[17] Ali Mohammed Javeed. Metrics for Requirements Engineering. 2006. Available from: www.cs.umu.se/education/examina/Rapporter/JaveedAli.pdf.

[18] Voas and Miller, "Software Testability: The New Verification". IEEE Software, Vol. 12(3), p. 17-28, 1995.

[19] J.M. Voas, "Object-Oriented Software Testability", In proceedings of International Conference on Achieving Quality in Software, January 1996.

[20] Bach, J. (1999). James Bach on risk-based testing. STQE Magazine, 1, 6.

[21] R.V. Binder, "Design for testability in object-oriented systems‖, Communications of the ACM Vol. 37(9), p. 87 - 101, 1994.

[22] ISO /IEC25010: Software engineering– system and software quality requirement and evaluation (SQuaRE)- system and software quality model; 2011.

[23] Esaki K. System quality requirement and evaluation, importance of application of the ISO/IEC 25000 series, Global Perspectives of Engineering Management.2013; 2(2):52-59

[24] Lewis, W. E. (2016). Software testing and continuous quality improvement. CRC press.

[25] Leach, R. J. (2016). Introduction to software engineering. CRC Press.

[26] L. Zhao, ―A new approach for software testability analysis‖, International Conference on Software Engineering, Proceeding of the 28th international conference on Software Engineering, Shanghai, pp. 985–988, 2006.

[27] 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.

[28] Drown DJ , Khoshgoftaar TM, Seiya N. Evaluation any sampling and software quality model of high assurance systems, IEEE Transaction on systems, Mean and Cybernetics, Part A: Systems and Human. 2009;39(5):1097-1107.

[29] Huda, M., Arya, Y.D.S. and Khan, M.H. (2015) Evaluating Effectiveness Factor of Object Oriented Design: A Testability Perspective. International Journal of Software Engineering & Applications (IJSEA), 6, 41-49. http://dx.doi.org/10.5121/ijsea.2015.6104

[30] Krruchtem P. The rational unified process: an introduction, Addison Wesley; 2000..

[31] Huda, M., Arya, Y.D.S. and Khan, M.H. (2015) Metric Based Testability Estimation Model for Object Oriented Design: Quality Perspective. Journal of Software Engineering and Applications, 8, 234-243. http://dx.doi.org/10.4236/jsea.2015.84024

[32] Robert V. Binder "Testing object-oriented systems: models, patterns, and tools", Addison-Wesley Longman Publishing Co., Inc., 1999.