IJARCCE

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

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 12, December 2015

Effectiveness Estimation of Object Oriented Software: A Revisit

Pooja Gupta[1], Dr. Namrata Dhanda[2]

Computer Science and Engineering, Goel Institute of Technology and Management, Lucknow, India[1]

Associate Professor and Head of Department, Department of CSE and IT, Goel Institute of Technology and Management, Lucknow, India[2]

Abstract: Effectiveness has always been an indefinable concept. Its truthful measurement or assessment is a complex exercise for the reason that of the various potential factors influencing effectiveness. It has been find out from systematic literature review that area researchers, quality controllers and industry personnel had made significant efforts to estimate software effectiveness but at the source code level. Calculating effectiveness at source code level directs to late arrival of desired information. An exact measure of software quality fully depends on effectiveness estimation. This paper shows the results of a systematic literature review conducted to collect related evidence on effectiveness estimation of object oriented software. In this revisit paper, our objective is to find the known complete and comprehensive software effectiveness estimation model and related framework for estimating the effectiveness of object oriented software at an initial stage of development life cycle.

Keywords: Effectiveness, Quality, Object Oriented properties, Quality Factors.

I. INTRODUCTION

At present software development industry, although knowing the benefits as well as necessity of producing quality software to end users, expected level of product's quality are becoming more demanding and vital. According to ISO 9126 'Software quality is the degree to which a system, component, or process meets customer or user needs or expectations'. Software quality is conformance to requirements and fitness for use, where product and corresponding artifacts meet the needs and expectations of the user [8]. With the increasing size of software applications, software development industry are lacking to produce the quality software and even several time a few product quality attributes are ignored [4].

In this present highly competitive software development industry and companies are frequently trying to achieve the release dead line that generally reduces the product quality. Therefore the delivered software product may not be appropriately checked for the probable defects. Software effectiveness is highly related to product quality and always plays a crucial role to produce best quality, high class and trustworthy software product within time and available budget [1] [2]. Effectiveness is one of the important notions in design and development for software programs and modules [3]. To design and develop a good quality and effective software product, effectiveness always play a significant role. Estimating software effectiveness at early stage in the software development life cycle may significantly reduce the overall development cost.

Effectiveness has been defined "fitness for use". Other definitions focus on the utility of the product or service.

I.I McCall's Quality Model

The McCall quality model given in the year 1977 has three major perspectives for defining and identifying the quality of a software product [14]: product revision (ability to undergo changes), product transition (adaptability to new environments) and product operations (its operation characteristics). In all these perspectives effectiveness plays important role to measure software quality.

I.II Boehm's Quality Model (1978)

Boehm's model is similar to the McCall Quality Model in that it also presents a hierarchical quality model structured around high-level characteristics, intermediate level characteristics, primitive characteristics - each of which contributes to the overall quality level. [9, 16]. In Boehm's Quality Model effectiveness uses as is utility of software product.

I.III Dromey's Quality Model

Dromey's is focusing on the relationship between the quality attributes and the sub-attributes, as well as attempting to connect software product properties with software quality attributes [6, 14, 18]. In this model effectiveness is defined as the products or services capability to meet customer expectations explicit or not, where software effectiveness is identified independent of any measurable characteristics.

II. RELATED WORK

Effectiveness can be predicted as soon as the system is specified. Freedman proposed domain effectiveness to address the problem of input inconsistency and output inconsistency, which involved use of the concepts of Observability and controllability [4, 19]. The ISO 9126 standard was based on the McCall and Boehm models [20, 21, 24].

Besides being structured in basically the same manner as these models ISO 9126 also includes effectiveness as a

parameter, as well as identifying both internal and external quality characteristics of software products. Dromey's is focusing on the relationship between the quality attributes and the sub-attributes, as well as attempting to connect software product properties with software quality attributes. Voas defined that effectiveness of a program is a prediction of the tendency for failures to be observed during random black-box testing when faults are present [25, 26]. They used DRR to indicate the inexplicit information loss, the bigger the DRR, the more information loss and so the effectiveness is smaller. In object-oriented software, Baudry took the number of class interactions in a UML class diagram as effectiveness measure to indicate the potential conflict that may occur in test, the more class interactions the lower the effectiveness [5]. Software structure has direct effect on test. Some complexity measures are assumed to imply the number of the test cases in term of structural coverage and so can indicate the effort to test the program to a certain degree. Richard defined effectiveness as the number of test cases that needed to satisfy certain test criteria, and computed it on the program control flow [7, 13]. Yeah accurately count the number of the test cases that needed to cover the program and introduced block normalization and structural normalization before the counting that based on data flow [10]. Abdullah used information transfer of between component and its context to indicate the effectiveness of certain component [11].

Fault/failure model reflects the behavioral characteristics of the software during testing. Reference [12] defined effectiveness as a prediction of the probability that existing faults will be revealed during testing given an arbitrary input selection criterion C. PIE is proposed to analysis the sensitivity of statement location by statically analysis its execution rate (E), infection rate (I) and propagation rate (P), which can indicate the effort to execute the test to gain certain confidence. But the computation of PIE is quite complex. Lin [14] reduced the estimation of the probability estimate by analyzing the semantic of the code and program structure. Huda [4] used one sample test suite to estimate the PIE rate. These method decreases the computation complexity with the cost of precision loss.

III. DESIGN PROPERTIES THAT INFLUENCES QUALITY

Object oriented technology direct the designers and developers what to take and what to avoid from. A Number of measures have been defined so far to assess object oriented design. There are a range of important themes of object oriented design that are recognized to be the basis of internal quality of object oriented software and help in the context of measurement. These themes extensively take account of encapsulation, coupling, cohesion and inheritance [16]. Encapsulation is the mechanism to hide the internal specification of an object and shows only the external interface. This means that all that is seen of an object is its interface, namely the operations we can perform on the object. Information hiding is the process of hiding all the information about the module unless it is specifically declared publicly". Information hiding gives rise to encapsulation in object oriented language [15].

Inheritance is an approach where an object acquires the characteristics from another object by sharing of attributes and operations among classes through their hierarchical relationship [22]. The new classes of objects that inherit much of their behavior from previously defined classes. Inheritance is a form of reuse that enable a process of development to define objects incrementally by reusing previously defined objects as the basis for new objects [23].

Polymorphism is an important concept that has a capability to build a flexible system. Polymorphism means, the ability to have several forms, which is to carry out different processing steps by the operations having same messages. Polymorphism allows the implementation of given operations, which are dependent on the object that contains the operations; an operation can be implemented in different ways in different classes [27].

The two more, most important design properties may be included, that have been generally used in designing of the software that is Coupling and Cohesion.

Coupling is the process to interact or communicate between two objects by passing messages. It refers to the degree of association from one object to another. It shows the relationship or interdependency between modules.

Coupling may assess the number of collaboration between classes or the number of messages passed between objects [28].

Cohesion is the process to measures the degree of connectivity among the elements of a single class or object [17]. It refers to the degree, to which the methods in a class are related to each other. The internal consistency occurs within the parts of the design, and it is focused on data that is encapsulated within an object and how the methods communicate with data to provide well bounded behavior [15].

IV. EFFECTIVENESS AT DESIGN PHASE

Quantification Programming methodology is based on objects that involved functions and procedures, this concept allows individual object to organize and group themselves together into class. That requires the effectiveness to be revealed because of the complex structure of object oriented development system because traditional testing approach is ineffective in this system. Practitioners incessantly support that effectiveness should be planned early in the design phase. So it is important to identify object oriented design artifacts to quantify effectiveness measures as early as possible in development life cycle. During identification of design factors which have positive impact on effectiveness estimation, a pragmatic view should be considered. If we consider all factors and measures then they become more complicated, ineffective and time consuming. So need to identify effectiveness factors and measures which affect the activity positively and directly [24]. In order to estimating effectiveness, 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 effectiveness. This process identifies object oriented design constructs that are used during design phase of development lifecycle and serve to define a variety of effectiveness factors. The contribution of each object oriented design characteristics is analyzed for improvement in design effectiveness.

Table1: OO Design Parameters Contributing in Effectiveness Estimation At Design Phase: A Critical Look

Design Parameters → / **Author/Study ↓**	**Cohesion**	**Coupling**	**Encapsulation**	**Inheritance**	**Abstraction**
MC Gregor et al. (1996)			√	√	
Bruce & Shi(1998)		√		√	
B.Pettichord(2002)		√			√
Baidry et al.(2002)		√			√
M Bruntik (2004)				√	
S.Mouchawrab (2005)	√	√		√	
I.Ahson et al.(2007)	√	√	√	√	
Nazir et al.(2005)	√	√	√	√	
Suhel et al.((2012)	√	√	√	√	√
Khan et al. (2012)	√	√	√	√	
Nikfard &Babak(2013)		√	√	√	
Abdullah et al.((2014)	√	√	√	√	
M. Huda et al.((2015)	√	√	√	√	

V. CONCLUSION

With growing complexity, pervasiveness and criticality of software, building reliable and quality end software is becoming more and more challenging. Moreover, the advancement in the software development process has been accelerated drastically in the last couple of decades. As a result, the complexity of applications and environments has been substantially increased and schedules have been pinched. Under these circumstances, software quality tends to suffer. In the face of intense competitive pressure, a comprehensive and rational strategy to achieve high effectiveness will be a strategic advantage-not a bottleneck. The foregoing analysis implies that effectiveness results from good Software Engineering practice and an effective software process.

Improving software effectiveness has become an important objective in order to reduce the number of defects that result from poorly designed software. Undoubtedly, effectiveness is a key factor to software quality and security, and plays an important role in delivering safe and quality software. It is an obvious fact that by estimating effectiveness early, a decision may be taken to incorporate changes at design phase. In order to fulfill the immense need of commonly accepted set of the factors affecting software effectiveness, an effort has been made in this paper in the form of a set of effectiveness factors early in design phase.

ACKNOWLEDGMENT

The heading of the Acknowledgment section and the References section must not be numbered.

Causal Productions wishes to acknowledge Michael Shell and other contributors for developing and maintaining the IEEE LaTeX style files which have been used in the preparation of this template.

REFERENCES

[1] Amin, A. and Moradi, S. (2013) A Hybrid Evaluation Framework of CMM and COBIT for Improving the Software Development Quality.

[2] Natasha Sharygina , James C. Browne, and Robert P. Kurshan, "A Formal Object-Oriented Analysis for Software: Design for Verification", 2011, pp:1-15

[3] 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.

[4] 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

[5] IEEE Press (1990) IEEE Standard Glossary of Software Engineering Technology. ANSI/IEEE Standard 610.12-1990.

[6] Nikolaos Tsantalis, Alexander Chatzigeorgiou, "Predicting the Probability of Change in Object-Oriented Systems", IEEE Transactions on Software Engineering, VOL. 31, NO. 7, July 2005, pp: 601-614.

[7] Abdullah, Dr, M. H. Khan, and Reena Srivastava. "Testability Measurement Model for Object Oriented Design (TMMOOD)". *International Journal of Computer Science & Information Technology (IJCSIT)*, Vol. 7, No 1, February 2015, DOI: 10.5121/ijcsit.2015.7115.

[8] ISO (2001) ISO/IEC 9126-1: Software Engineering—Product Quality—Part-1: Quality Model. Geneva.

[9] Stephanie Gaudan , Gilles Motet and Guillaume Auriol , "A new structural complexity metrics applied to Object Oriented design assessment", http://www.lesia.insa-toulouse.fr/~motet/papers/2007_ISSRE_GMA.pdf.

[10] Everald E. Mills, "Software Metrics", SEI Curriculum Module SEI-CM-12-1.1, Software Engineering Institute, Dec 1988, pp: 1-43.
[11] Haifeng Li Minyan Lu Qiuying Li , "Software Metrics Selecting Method Based on Analytic Hierarchy Process", Sixth International Conference on Quality Software, 2006. QSIC 2006, 27-28 Oct. 2006, pp: 337 – 346, ISSN: 1550-6002, ISBN: 0-7695-2718-3.
[12] 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
[13] Offutt, R. and R. Alexander, (2001): A fault Model for Subtype Inheritance and Polymorphism. In 12th International Symposium, Software Reliability Engineering, Nov 27-30, 2001, IEEE, pp. 84-93.
[14] Jagdish Bansiya, "A Hierarchical Model for Object Oriented Design Quality Assessment", IEEE Transaction of Software Engineering, Volume 28, No. 1, January 2002, and pp: 4-17
[15] Binder, R.V. (1994) Design for Testability in Object-Oriented Systems. *Communications of the ACM*, **37**, 87-101. http://dx.doi.org/10.1145/182987.184077.
[16] 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.
[17] Yong Cao Qingxin Zhu. Improved metrics for encapsulation based on information hiding. DOI: 10.1109/ICYCS.2008.76, The 9th International Conference for Young Computer Scientists, IEEE computer society 2008, p: 742-724.
[18] Huda, M., Arya, Y.D.S. and Khan, M.H. (2015) Quantifying Reusability of Object Oriented Design: A Testability Perspective. *Journal of Software Engineering and Applications*, **8**, 175-183. http://dx.doi.org/10.4236/jsea.2015.84018
[19] Sch aril N., Black Andrew P., Ducasse S. Object oriented Encapsulation for Dynamically Typed Languages. OOPSLA 2004, ACM, pp: 130–139.
[20] 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.
[21] Usha Chhillar, Shuchita Bhasin , " A New Weighted Composite Complexity Measure for Object-Oriented Systems", International Journal of Information and Communication Technology Research Volume 1 No. 3, July 2011,pp: 101-108, ISSN-2223-4985.
[22] Huda, M., Arya, Y.D.S. and Khan, M.H. (2015) Testability Quantification Framework of Object Oriented Software: A New Perspective. *International Journal of Advanced Research in Computer and Communication Engineering*, **4**, 298-302. http://dx.doi.org/10.17148/IJARCCE.2015.4168
[23] Dromey, R.G.: A Model for Software Product Quality. IEEE Transaction on Software Engineering 21(2), 146–162 (1995).
[24] Huda, M., Arya, Y. D. S., Khan, M. H., & Dima, M. O. Scientific Research.
[25] 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
[26] Fiondella, L.; Gokhale, S.S., "Software quality model with bathtub-shaped fault detection rate" Reliability and Maintainability Symposium (RAMS), 2011 Proceedings - Annual , 24-27 Jan. 2011,pp: 1 – 6, ISBN: 978-1-4244-8857-5.
[27] Huda, M., Arya, Y.D.S. and Khan, M.H. (2014) Measuring Testability of Object Oriented Design: A Systematic Review. *International Journal of Scientific Engineering and Technology* (*IJSET*), **3**, 1313-1319.
[28] Mohan, K.K.; Verma, A.K.; Srividya, A., "Software effectiveness estimation through black box and white box testing at prototype level ", 2nd International Conference on Reliability, Safety and Hazard (ICRESH), 14-16 Dec. 2010, pp: 517 - 522, ISBN: 978-1-4244-8344-0.