RESEARCH REVIEW JOURNALS
Volume-04
Issue-06
June-2019

ISSN: 2455-3085 (Online)
RESEARCH REVIEW International Journal of Multidisciplinary
www.rrjournals.com[UGC Listed Journal]

Study of Dependence Matrix for Assessment of Object-Oriented Software Design Quality

[1]Mukesh Kumar Gupta & [2]Mr. Tarun Dalal

[1]Department of Computer Science & Engineering, CBS Group of Institutions, MDU, Rohtak, Haryana (India)
[2]Assistant Professor, Dept of CSE, CBS Group of Institutions, MDU, Rohtak, Haryana (India)

ARTICLE DETAILS

Article History
Published Online: 12 June 2019

Keywords
Object-Oriented, Software design quality

ABSTRACT

The purpose of this paper is to evaluate if software metrics can be used in software development process to improve the design quality of the software system. One of the earliest software product quality models was suggested by McCall and his colleagues. It defines software product quality as a hierarchy of factors, criteria and metrics. ISO developed a standard for software product quality, ISO 9126, which is on the same line. Problem with these models is that they are vague in their definition of lower level attributes and metrics needed to attain a quantitative assessment of product quality. Recently, a framework for building product based quality models has been developed by Dromy which addresses some of the problems of earlier models. This paper takes up two approaches in parallel to evaluate the quality of Object-Oriented Software System. The first approach extends the concept of dependence matrix of modular programming to OO Systems. Coupling measure is calculated comprehensively for all possible combination types of interactions among classes. These interactions may occur due to inheritance, friendship or containership (whole-part) relationships and may be different types e.g. class-attribute interaction, class-method interaction and method-method interaction. Interactions are also distinguished by locus of impact: i.e. whether the impact of change flows towards a class (import) or away from class (export). Next, all five of cohesions are considered to obtain the cohesion measure. Using these two, first order dependence matrix and complete order dependence matrix are calculated which gives expected change measure and overall design measure.

1. Introduction

Many of the Quality Metrics Models available for Object Oriented Software Analysis can be applied only after the product is complete or nearly complete. They rely upon information extracted from implementation of the product. This provides information too late to help in improving the software design and codes. Thus, there is a need for metrics and models that can be applied at early stages of development (i.e. requirement and design level). The proposed paper addresses this long needed requirement of software industry as its fulfillment shall empower the developers with effective design plans and thus quality end product [1].

The need for software quality has been extensively continuous because of the increasing in the community need for software in all aspects of life. The importance of providing the highest quality standards is no longer an advantage, but it is necessary for companies to be successful and competitive [2]. Therefore, there is a unified agreement on the need for software quality, and a number of software quality models and software measurements appeared to solve this problem [2]. Everything in life, if it can be measured, gives the ability to deal with it smoothly, so the importance of software measurements provides indications on the quality and strength of software [3, 4]. In addition, the level of development and improvement in the quality models to show the enhancement and defects in the software from one version to another is an important area to be studied [5]. The concept of the clean code is used to distinguish the quality of the code in terms of readability, understanding, structure and complexity [6-9]. All these characteristics are difficult to be identified by depending only on the experiences of programmers and developers [10]. It is necessary to come out with approaches that solve these problems more effectively than the experience of programmers, which may not be available at any time or expensive for the software developer. The contribution of this paper is proposing a general software quality model to give more flexibility and control during the development of software by making the proposed quality model work on three levels during the analysis of the software product. The proposed model can also display visual chart indictors with recommendations in case some expected defects in software product are found. For this purpose, a tool has been built to measure the quality of the software with an object-oriented programming that measures the quality at three levels: package, class, and function (Method). At the package level, the tool gives the package details in terms of complexity measures and threshold limits on the aspect and quality attributes of Bansiya quality model [4] normalized by threshold value to explain high level of abstraction in software. The Class level has been treated with the extraction of special measures that can be extracted from the class only. With comparison to the appropriate threshold limits, the quality of this class can be judged. At low level of source code (Function level), a quality attribute has been proposed for clean code, which is a set of parameters that are only within the function limits, by which the function can be classified as either having the clean programming code attribute or not. Moreover, the proposed software quality model suggested using of visual

color alerts, and recommendations systems in order to help software engineer in the process of evaluating software product [11.

2. Literature Review

This section reviews the previous works related to the most widely used software quality model to determine the ambiguity of the concept of software quality and the scope of those models in real applications developments. In addition, these works focus on the complexity of the source code (Source Code Complexity Metrics)in terms of the use mechanisms, analysis, and the appropriate threshold for each metrics. In 2006, Gitter conducted a study on how to apply the Bansiya software quality model to evaluate the development of 19 Azureus versions by making comparisons using object oriented metrics rather than the metrics proposed by Bansiya and his colleagues. The researcher demonstrated the ability of this model to track the evolution of design quality in several versions by providing access to important information in the internal life of the software.

This information can support design decisions at higher levels of abstraction. His proposed model may require additional inputs to cover the highest levels of abstraction to assess all aspects of the quality model at this level [10].Another study by Panfowski (2008) presented a new assessment of software product quality, which focused on assessing the quality of the external features of the software product, which means evaluating the behavior of the software product when implemented. In addition, the study focused on the development of the quality model (ISO / IEC 9126) at the level of software metrics.

The study relied on seven samples of the software product and evaluated them using ISO / IEC 9126-2 quality model. In his work, Panfowski concluded that external product quality attributes are an area or category that can be adopted, and that the metrics provided by ISO / IEC 9126-2 can be considered as a starting point for the definition of standards, but are not ready to use in their present form. The metrics of the software product need to be more adapted to show better information [10]. Borgherth (2008) discussed the method of code profiling by using a static analysis. The study was done on (19) industrial samples and (37) samples of students' programs. He has analyzed software samples through software metrics.

The results of this study indicated that the code pattern could be a useful technique for rapid program comparisons and quality observation in the field of industrial application and education [12]. Moreover, Bhatti (2010) explored the area occupied by the software metrics. He used a QA-C tool to measure software metrics automatically on the code written in C programming language through expressing the association between software metrics and the complexity of the source code. He attempted to demonstrate the values of these metrics graphically only, without considering the quality features and threshold limits relationship [13].Another work in 2010 is the impact of code complexity and usability, either in monitoring software complexity during development, or in evaluating the complexity of legacy software.

The researchers of this work, Goran and Dahiden proposed a new coupling metrics (Ecoup), and introduced the Java met tool, which works in a static analysis of programs written in Java with respect to coupling, flow control, complexity and coherence [14].In the same year, Chandra et al. proposed the use of Object Oriented metrics that introduced by Chidamber and Kemerer (1994)[15] to assess program quality at the class level. The proposed tool can be used to verify the class design conforms to the design specifications of the Object Oriented programming, through using the threshold for each metric [16]. The following is a summary of the most important software quality models: 1) McCall's Factors-Criteria (FCM) model presented by McCall in 1977.

The McCall model is the first model of quality [1, 10]. 2) ISO / IEC 9126: 2001, which was submitted by the International Organization for Standardization (ISO) in 1991, and has developed six quality metrics. This model was updated in 2001, Quality ISO / IEC 9126: 2001 [10, 13,]. 3) Dormey's Quality Model, presented by Jeff Dormy (1998) as a quality assessment model, by analyzing the quality of program components through measuring concrete quality characteristics [4, 10]. 4) Boehm Software Quality Model presented by the scientist Erwin Bohm in 1978. This model seeks to determine the quality of the program through a predefined set of metrics and measures [9]. 5) FURPS Quality Model, introduced by Grady Robert and Hewlett-Packard in 1987. This model focuses on the analysis of quality characteristics in two categories of requirements: functional requirements and non-functional requirements [11, 17]. 6) Bansiya Quality Model, proposed by Bansiya in 2002. This model focuses on the quality of Object Oriented Design (QMOOD). It uses the source code metrics extract directly from the software source code to give the quality attribute through the use of mathematical equations [1, 10].

The Bansiya Quality model gives a way to assign source code measurements to higher abstraction levels [10]. Although the experiment results in this model were acceptable, the use of new and non-standard measures in OOP metrics makes this model not widely used, and this is why Gitter in [10] tried to change the measures used in this model to the stranded OOP metrics so that the model becomes more dependable. For this reason, Bansiya model with stranded OOP metrics was used at the software packages analysis stage in the proposed model. Standard OOP metrics used in the proposed model can be viewed in [6, 10]. Furthermore, the metrics threshold values that used for the recommendations and alerts are selected of this proposed model due to their usage in the following references and full described in [9, 16], which are the reason of the selection.

3. Factor-Strategy Model

Based on the detection strategy mechanism we propose a new type of quality model, called Factor-Strategy (FS). This approach is intended to improve the FCM paradigm with respect to the two major drawbacks. In Figure 2 we illustrate the concept of a Factor-Strategy model. FS models still use a deco positional approach, but after decomposing quality in

factors, these factors are not anymore associated directly with a bunch of numbers, which proved to be of a low relevance for an engineer. Instead, quality factors are now expressed and evaluated in terms of detection strategies, which are the quantified expressions of the good-style design rules for the object-oriented paradigm. Therefore we may state in more abstract terms that in a Factor-Strategy model, quality is expressed in terms of principles, rules and guidelines of a programming paradigm. The set of detection strategies defined in the context of a FS quality model encapsulate therefore the knowledge box of good design for the given paradigm. The larger the knowledge-box, the more accurate the quality assessment is. In our case the detection strategies are defined for the object-oriented paradigm, and thus in the right side of Figure 2 we depicted a sample of a knowledge-box of object-oriented design. The knowledge-box, as such, is indispensable for any quality model. Although not visible at first sight, it is also present in the FCM models. The knowledge box is not obvious in the FCM approach because of its implicit character, while it becomes explicit in the FS model.

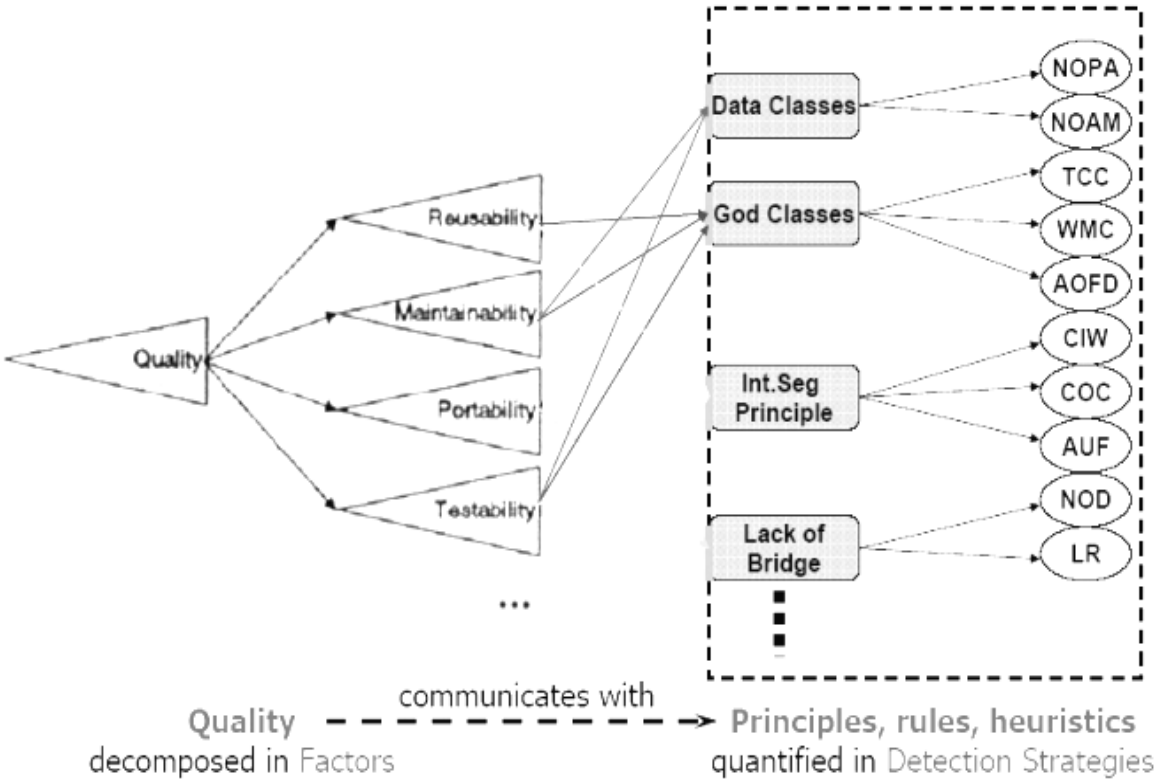

Figure 1: Factor-Strategy model.

4. Construction methodology

The main issue in constructing any type of quality model is how to build the association between the higher and the lower levels of the model e.g., in building a FCM model we are concerned with associating the quality factors with the proper criterion, or how to choose the metrics for a given criterion. As the Factor-Strategy models are also based on a decompositional approach the association is still the relevant issue. Based on the previous considerations, we identify two distinct aspects on this matter of association: a semantical and a computational aspect.

- The semantical aspect must sustain the validity of choosing a particular decomposition for a higher-level element into lower-level ones. In other words, it must explain the rationale behind the association i.e., why and how do we choose a particular decomposition for a higher-level element of the model?
- The computational aspect must tackle the issue of quantifying the association i.e., how the quality score for the higher-level element is to be computed from the quality scores of the lower-level elements associated with it.

Obviously, we must first define an association that “stands” semantically, and only then the focus must be set to finding the association formula that quantifies it. The association formula must reflect the “participation level” of each lower level element within the higher-level aspect of quality. In constructing a Factor-Strategy model there are two association that must be done: the decomposition of the quality goal in quality factors and the association between these factors and detection strategies that can detect design flaws that affect the given quality factor.

Decomposition of the Quality Goal in Factors

There are two possible approaches to address the semantical aspect of the association between a quality goal and a set of quality factors: we can either rely on a predefined decomposition found in the literature or go for a user-defined one. The former option has the advantage of a wider acceptance, while the latter is more flexible and adaptable to the particular investigation needs. In this paper we rely on an

existing and widely accepted decomposition i.e., the one found in the ISO9126 standard [12]. For the general case we recommend using a hybrid solution: start from a predefined decomposition found in the literature that comes closest to your ideal model and then slightly customize it until it matches your perspective on quality. This association is orthogonal to the programming paradigm used for the design and implementation of the system. The decompositions found in literature, in spite of many differences, keep the higher level of quality decomposition 3, abstract enough to make it independent of the development paradigm. As a consequence, in a FS model the decomposition of a quality goal in factors is not different in any aspect to that found in the FCM approach. Therefore, the computational aspect of this association does not raise additional discussions at the conceptual level.

Association of Factors with Detection Strategies

Detection strategies used in FS models capture deviations of a design from design rules and guidelines. In [18] we have described in detail the process of transforming informal design rules into detection strategies. The process of identifying the metrics needed to a quantify a given rule and the way the metrics are correlated to capture that precise aspect is done very much like in the ”Goal-Question-Metric” approach [2]. The authors of such good-design rules implicitly or explicitly relate them to quality factors [19, 10], or to abstract design principles [13] (e.g., abstraction, modularity, simplicity) that can be easily mapped to quality factors. Therefore, the semantical aspect of the association between quality factors and detection strategies becomes self evident in the FS approach.

5. Results and discussions

From the experimentation carried out on a variety of sample programs, it can be observed that although both the approaches for determining strength/weakness of the program are quite different, they are fundamentally related to some extent. From the figure 2 it can be observed that if we move towards a higher ODM value i.e. a weaker design, and then Overall design quality decreases more or less linearly indicating a weaker design. Similarly, from the figure 3, it may be concluded that if the strength of design increases the value of Overall design quality also increased. However, this is a general trend and some exceptions have also been observed during experimentation with the sample programs. These aberrations may be accounted to some attributes in assessment of overall design quality which could not be incorporated while calculating the complete order dependency matrix e.g. Polymorphism.

Table 1: Experiment results for some C++ program source inputs

S. No.	Program Name	Overall Design Measure (ODM)	Strength of Design	Overall Design Quality
1.	C++ Unit Test Framework	0.8	0.2	4.414
2.	Emp.cpp	0.492	0.508	14.404
3.	Multiple.cpp	0.839	0.161	7.108
4.	Soft.cpp	0.74	0.26	4.952
5.	Slot.cpp	0.561	0.439	9.114

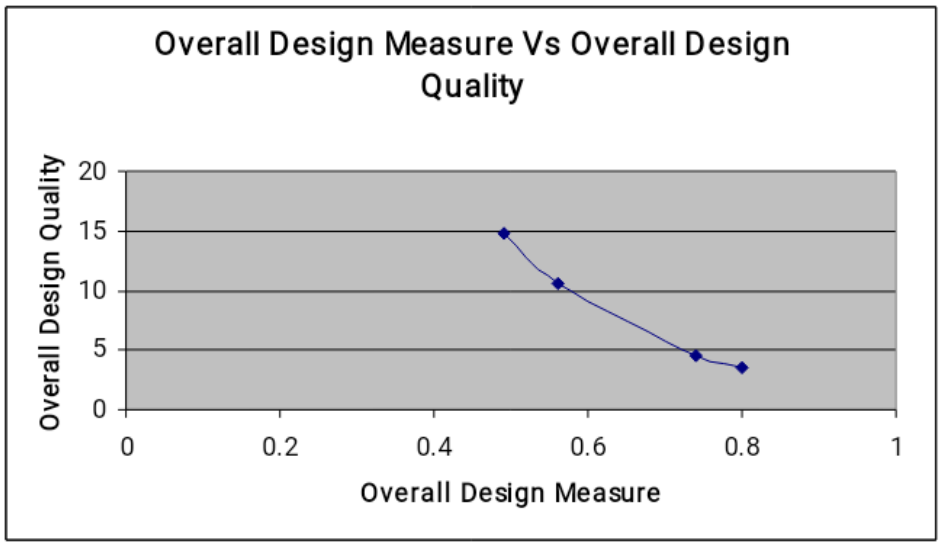

Figure 2: Overall Design Measure Vs Overall Design Quality

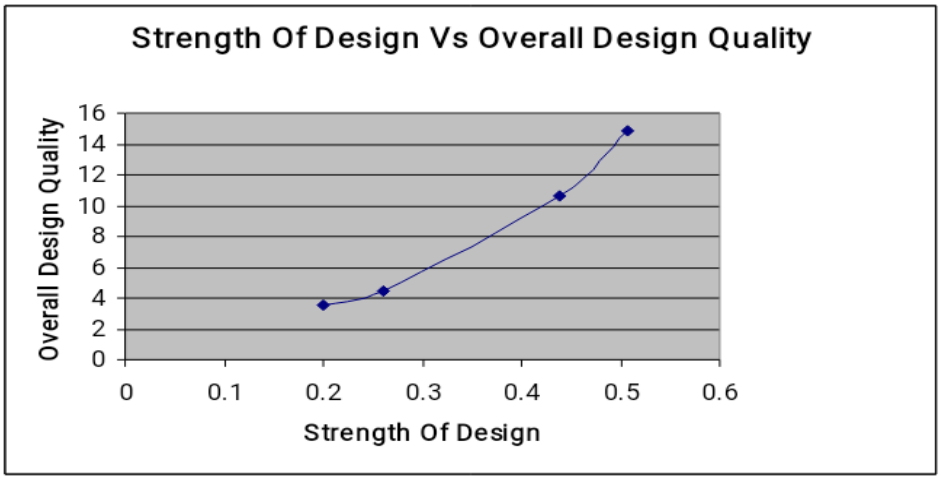

Figure 3: Strength of Design Vs Overall Design Quality

6. Conclusions

Computation of overall design measure and strength of design for OO Systems using the concept of dependency metrics is an exhaustive process. Since the computation of coupling measure for OO systems involves several facets including inheritance, most of the design level OO quality attributes are incorporated in it. Thus, it is reasonable to validate this model using the hierarchical model for the design quality assessment of OO Systems. The later is already validated in [1] for different versions of MFC (Microsoft Foundation Classes) and Object Windows Library (OWL) which have been written for similar and consistent requirements.

The results confirm that Coupling and Cohesion are the two metrics that play major and decisive role in determining the overall design quality of the OO Software Systems. They can easily be gathered from the structure design of the OO System. They can be also used in software development process to improve the design quality of the object-oriented software system e.g. using the class fault probability obtained from coupling matrix, more attention may be given to classes with higher fault probability.

References

[1] Jagdish Bansiya and Carl G. Davis “A hierarchical Model for Object-Oriented Design Quality Assessment”, IEEE Transaction on Software Engineering, vol. 28, No. 1, January 2002

[2] Lionel Briand, Prem Devanbu and Walcelio Melo, “An Investigation into Coupling Measures for C++”.

[3] G. R. Dromy, “ A Model for Software Product Quality”, IEEE Transaction on Software Engineering, vol. 21, no. 2, Feb 1995

[4] S.R. Chidamber and C.F. Kemerer, “A Metric Suite for Object Oriented Design”, IEEE Transaction on Software Engineering, Vol. 20, No. 6, pp. 478-493, June 1994

[5] Ralf Reibing, “Towards a model for Object Oriented Design Measurement”, Institute of Computer Science, University of Stuttgart, Stuttgart, Germany.

[6] Aggarwal, K.K. and Yogesh Singh, 2005. Software Engineering. Rev. Sec. Edn., New Age International Publisher.

[7] *Software Engineering Standards*, 1994 Edition, IEEE.

[8] Aggarwal, K.K., Yogesh Singh, Jitender Kumar Chhabra, 2003. A multiple parameter software complexity measure.

[9] Aggarwal, K.K. and Yogesh Singh, 1994. A modified approach for software science measures. ACM SIGSOFT Software engineering Notes, USA.

[10] CROSBY, P. ,1979. QUALITY IS FREE. NEW YORK: MC GRAW-HILL. (CH.24).

[11] Vibhash Yadav, Prof. Raghuraj Singh and Prabhat Verma “Use of Dependence Matrix for Assessment of Object Oriented S/W Design Quality” IEEE International Conference (IACC09), Thapar University, Patiala on 6-7 march 2009.

[12] Vibhash Yadav, Prof. Raghuraj Singh and N.N. Das “Key Factors for increase of Design Quality in Rule Based OO Software” accepted for presentation in International Conference on Innovative Technologies (ICIT09) in 18-19 June 2009.

[13] Preeti Baser, Dr. Jatinderkumar R. Saini “A Comparative Analysis of Various Clustering Techniques used for Very Large Datasets”

[14] Kehar Singh, Dimple Malik and Naveen Sharma "Evolving limitations in K-means algorithm in data mining and their removal" IJCEM International Journal of Computational Engineering & Management, Vol. 12, April 2011,

[15] Keshav Sanse, Meena Sharma ”Clustering methods for Big data analysis” International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 3, March 2015

[16] Btissam Zerhari, Ayoub Ait Lahcen, Salma Mouline “Big Data Clustering: Algorithms and Challenges” Research Gate Conference Paper May 2015

[17] Yi Wang, Qixin Chen, Chongqing Kang, Qing Xia “Clustering of Electricity Consumption Behavior Dynamics Toward Big Data Applications “ in IEEE transactions on smart grid, vol. 7, no. 5, september 2016.

[18] Apache Hive. Available at http://hive.apache.org

[19] http://blogs.worldbank.org/voices/meet-winners-and-finalists-firstwbg-big-data-innovation-challenge.

