International Journal of Innovation and Scientific Research
ISSN 2351-8014 Vol. 13 No. 2 Feb. 2015, pp. 515-522

http://www.ijisr.issr-journals.org/

Framework to Evaluate Quality of Object Oriented Design

Ayat Al-lawatiya and Santhosh John

Dept. of Computing, MEC,
Knowledge Oasis Muscat, Al Rusayl,
Sultanate of Oman

ABSTRACT: In recent years, the usage of the object oriented paradigm in software development has increased. Consequently, by using object oriented software, new elements have been added to software development process. The design phase is the backbone to develop any object oriented software. Therefore, the object oriented metrics are used to measure the quality of design.

This paper describes a framework for evaluating the object oriented design. The framework relates the design properties such as: Encapsulation, Coupling, Cohesion, Abstraction, Complexity, Composition, Messaging, Inheritance, Hierarchies, and Polymorphism to high level quality attributes such as Reusability, Effectiveness, Extendibility, Understandability, Correctness, and Flexibility. Meanwhile, the design properties in classes are defined to be assessed by using a suit of object oriented design metrics. An empirical data is collected from four case studies to calculate the metrics and then apply them to calculate the quality properties.

KEYWORDS: Framework; Quality Attributes; Object Oriented Design Properties; Object Oriented Metrics.

1 INTRODUCTION

As the software systems are being used in different critical areas of industry, the quality of the software is becoming very critical to the success of business and human safety. Furthermore, the quality of the software is vital in these days to minimize the defects in the software development and to decrease the cost and efforts of maintenance.

It must be acknowledge that there is no ideal software. However, Object Oriented system has become very popular in software development environment because of its reusability, modularity and extensibility. It has become easy to maintain, fast to develop, and clear moving from real world entities to the object system.

One of the most important factors that affect the quality of the object oriented software is the design structure. That's why, evaluation of the quality of Object Oriented design is an essential part of software environment at this stage since it does not make exception from errors and faults it is prone to. Recently, there has been enough research to develop and empirically validate metrics for Object Oriented design quality. However, the importance of quality of the software is to apply it in its early stage to avoid such costs and efforts as presented in figure 1, s the latest phase are more expensive to correct the defects in the software. Furthermore, assessing the software quality at the design phase is more beneficial in guiding the development effort in subsequent phases. In other words, the analysis of the design phase is a basis for the software implementation.

Corresponding Author: Ayat Al-lawatiya

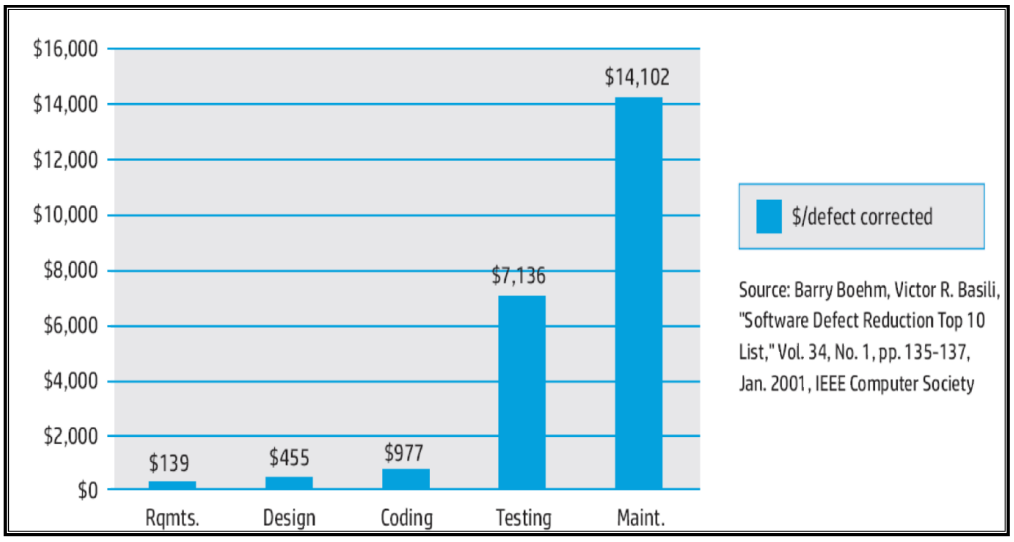

Figure 1: The cost of correcting defects during the software development life cycle (Eliminating IT waste from software defects lowering the cost of application quality n.d.)

The design of the software does not take that much effort during developing the software. However, if there is any mistake or defect in the design, the cost of maintenance will be 90% higher than the total cost of the life cycle of software development (Akaikine 1997). The main step in evaluating the quality of Object oriented software during the development life cycle software is measuring the quality at the design phase. Assessing the quality of design is done by using appropriate metrics and evaluation techniques. There are many metrics suits that were proposed to measure the quality of object oriented design like: CK metrics, QMOOD suit metrics, Lorenz and Kid metrics and MOOD metrics. These metrics significantly reduce rework during and after implementation to design effective test plans.

2 RESEARCH PROBLEM AND OBJECTIVES:

The scope of this research is to propose a framework to evaluate the quality of Object Oriented design phase. This will lead to predict some software quality characteristics based on the connection between the quality attributes and design properties by using object oriented design metrics. The reason behind defining this scope is to help developers reduce the challenges that object oriented software development entities are facing during the Software Development Life Cycle (SDLC). This, as a result, will reduce software maintenance efforts, over-budgeting or delays. It will also help developers fix problems and eliminate unwanted complexities in the early stage of development cycle.

Today, many metrics and quality models are available for assessing the quality of Object oriented. Most of these can be applied at the design phase of software development when the product is completed or nearly completed. These approaches analyze the software code to fetch the software metrics used in determining its quality.

This makes it difficult to streamline the design for its improvement. Therefore, there is a need to apply the evaluation of the quality of software at an early design stage of the software development process. Detection of a faulty design at later stages of software development bears a heavy price in terms of effort and cost (Yadav and Singh 2013). Thus, there is a need for models which could make an assessment of the software quality during the design phase of software development (Bansiya and Davis 2002).

The objective of the research is to propose a framework to evaluate the quality of the object oriented design. This report is organized in the following way. The Chapter П describes framework development, Chapter Ш is about the proposed framework and Chapter 5 includes the conclusion and suggestion for future research.

3 FRAMEWORK DEVELOPMENT:

The framework has many principal elements like:

- Identifying a set of high- level quality attributes.
- Identifying object oriented design properties.
- Identifying object oriented design metrics.

- Assigning design metrics to design properties.
- Linking design properties to quality attributes.

3.1 IDENTIFYING THE QUALITY ATTRIBUTES:

The combination of quality attributes between different object oriented models like: McCall's Model, Boehm's Model and ISO 9126 Model were selected as the initial set of quality attributes in the proposed framework as shown in the following table.

Table 1: comparison between quality attribute of four common models of qualities attributes:

No	software quality attributes	McCall's model	Bohem	ISO 9126	A hierarchical model for object oriented design quality assessment
1	Portability	✓	✓	✓	
2	Reusability	✓	✓		✓
3	Maintainability	✓	✓	✓	
4	Flexibility	✓	✓		✓
5	Testability	✓		Maintainability	
6	Interoperability	✓			
7	Correctness	✓	✓	Maintainability	
8	Reliability	✓	✓	✓	
9	Efficiency	✓	✓	✓	✓
10	Integrity	✓	✓		
11	Usability	✓	✓	✓	
12	Functionality		✓	✓	✓
13	Understandability		✓		✓
14	Extendibility		✓		✓
15	Validity		✓	Maintainability	
16	Generality		✓		
17	Clarity		✓		
18	Modifiability		✓	Maintainability	
19	Documentation		✓		
20	Resilience		✓		
21	Economy		✓		
22	Security				
23	Fault-Proneness				

This set of attributes was reviewed and the common attributes between the compared models were chosen and are the following attributes: portability, reusability, maintainability, correctness, efficiency, usability and functionality. After that, this set of attributes was individually reviewed to distinguish if they contribute towards defining design quality and include all aspects of the design quality.

The attribute "usability" was excluded since it is more appropriate to the context of software implementation rather than design phase. While, the term "efficiency" was replaced by term "effectiveness" which is more appropriate to describe the quality in the design stage. Furthermore, the term "extendibility" is a better reflection of characteristic in the design phase rather than using term "portability" which is more suitable to use in the quality of software implementation. The term "maintainability" was replaced by term "understandability" which is more specified for design quality features.

In order to achieve an important goal in adopting the object oriented approach in design and implementation, the following characteristics must be considered like reliability flexibility, and adaptability in the development process. Reuse of the development at all levels will achieve this objective and that's why the attribute "reusability" is necessary in the design stage. In addition, the attribute "flexibility" is an important characteristic in the design phase and will therefore be included as design quality attribute. Furthermore, the term "correctness" is an important design quality attribute since it allows the designer to modify the structure of the system to achieve the quality at that stage.

Thus, the initial set of design quality attributes in the proposed framework is: "The proposed set of attributes in this framework not exclusive and can be changed under any changes in the goals or objectives."

Table 2: The proposed quality attributes definition:

Quality Attribute	Definition
Functionality	The capability of the design class to provide functions which meet stated and implied needs through the public interface.
Effectiveness	The capability of software design to provide appropriate functionality and behavior using object oriented design concept.
Extendibility	Reflects the presence and usage of properties in the design which allocate any additional requirements in the design.
Understandability	The characteristics of the design that enable it to be easily learned and this related to the complexity of the design structure.
Reusability	Extent to which a design can be reused in to other problem without significant effort.
Correctness	Extent to which a program satisfies its specification and fulfills the customer's mission objectives.
Flexibility	The characteristics of design that allow the design to be adapted to provide functionality.

3.2 IDENTIFYING OBJECT ORIENTED DESIGN PROPERTIES:

Design properties can be evaluated directly by testing the internal and external structure, relationship, and functionality of the design components with its attributes, methods and classes. QMOOD represents the design properties for both structural, object oriented development and object oriented paradigm.

Table3: The design properties definition for proposed framework:

Design properties	Description
Design size	Measuring the number of classes used in the design.
Hierarchies	Representing the generalization-specialization concepts in the design. It is counting the number of non-inherited classes that have children in the design.
Abstraction	A measure of generalization aspect of design.
Encapsulation	It is a characteristic for designing classes by defining them as private to avoid access to the attribute declaration.
Cohesion	Evaluating the relationship of attributes and methods in class.
Coupling	Measuring the number of other objects that can be accessed by an object in order to complete the function correctly.
Composition	Measuring the aggregation relationship in an object oriented design.
Inheritance	Measuring the relationship between classes which is related to the level of nesting of classes in inheritance hierarchy.
Messaging	Measuring the number of services provided by class and counting the number of public methods which are services to other classes.
Complexity	Measuring the degree of understanding the relationship between the internal and external structure of classes

3.3 IDENTIFYING OBJECT ORIENTED DESIGN METRICS:

The need to measure the software is becoming very important leading to new software measures. There are different metrics that have been proposed with different aspects like coupling, cohesion, inheritance, information hiding, and

polymorphism. However, it is often difficult to determine which metric is more useful in which area. As a consequence, selecting a measure of the object oriented systems becomes very difficult for project managers and practitioners.

Each design properties identified in the proposed framework represent an attribute of a design that can be assessed by using defined metrics in the design phase. There are several metrics that are used to assess the design properties such abstraction, inheritance and messaging. The combination of metrics between CK, MOOD and QMOOD are the following:

As Khan et all examined MOOD.MOOSE, EMOOSE and QMOOD of metrics. They observed that QMOOD set of metrics is a combination between design and code. However, MOOSE set doesn't purely relate to the design. On the other hand, EMOOSE suit metrics do not relate to design phase as explained in (Khan, Mustafa and Ashon 2006). Based on this discussion, this research will be concentrate on the CK, MOOD and QMOOD. Response for class (RFC) was excluded in the proposed framework since it depends on the methods calling from outside the class which means it depends on the communication between classes which is more suitable for the implementing phase rather than design phase.

Table 4: Design Quality Metrics:

No	Design properties	Metric	Name of the metric	Descriptions
1	Design Size	DSC	Design size in class	Counting the number of classes in the design.
2	Hierarchies	NOH	Number of Hierarchies	Counting the number of class hierarchies in the design.
3	Encapsulation	MHF	Method Hiding Factor	Measuring the number of hidden methods
		AHF	Attribute Hiding Factor	Measuring the number of hidden attributes
		DAM	Data access Metrics	Measuring the ratio of the number of private attributes to the total number of attributes declared in the class.
4	Coupling	DCC	Direct Class Coupling	Counting the different number of classes related to that a class directly.
5	Inheritance	MIF	Method Inheritance Factor	Measuring the number of inherited methods
		AIF	Attribute Inheritance Factor	Measuring the number of inherited attributes
		DIT	Depth of Inheritance Tree	Counting the number of ancestor classes
		NOC	Number Children	Counting the number of subclasses that are going to inherit information from the parent
		MFA	Measure of Functional Abstraction	Measuring the ratio of the number of methods inherited by a class to the total number of methods can be accessed by member methods of the class.
6	Abstraction	ACA	Average count of ancestor	Counting the average number of classes from which a class inherits information
		ANA	Average Number of Ancestors	Measuring the average number of classes from which a class inherits information
7	Cohesion	CAM	Cohesion among methods in class	It computes the relatedness between methods of a class. It is summation of the intersection of methods' parameters with maximum independent set of all parameter types in the class
		LCOM	Lack of cohesion in method	Measuring the amount of cohesiveness present and the designing of the system and complexity of a class
8	Messaging	CF	Coupling Factor	Counting the number of classes that non-inherited coupled with other classes.
		CIS	Class Interface Size	Counting the number of public method in the class.

9	Complexity	WMC	Weighted Method per Class	Measuring the complexity of a class by measuring the number of methods
		NOM	Number of Methods	Counting the number of all methods in the class
10	Composition	MOA	Measure of aggregation	Counting the number of declared data which is user defined class

3.4 MAPPING QUALITY CARRYING COMPONENT PROPERTIES TO DESIGN PROPERTIES:

Table 5: Assigning design metrics to design properties:

No	Design properties	Metric	Name of the metric
1	Design Size	DSC	Design size in class
2	Hierarchies	NOH	Number of Hierarchies
3	Encapsulation	MHF	Method Hiding Factor
		AHF	Attribute Hiding Factor
		DAM	Data access Metrics
4	Coupling	DCC	Direct Class Coupling
5	Inheritance	MIF	Method Inheritance Factor
		AIF	Attribute Inheritance Factor
		DIT	Depth of Inheritance Tree
		NOC	Number Children
		MFA	Measure of Functional Abstraction
6	Abstraction	ACA	Average count of ancestor
		ANA	Average Number of Ancestors
7	Cohesion	CAM	Cohesion among methods in class
		LCOM	Lack of cohesion in method
8	Messaging	CF	Coupling Factor
		RFC	Response for a class
		CIS	Class Interface Size
9	Complexity	WMC	Weighted Method per Class
		NOM	Number of Methods
10	Composition	MOA	Measure of aggregation

3.5 LINKING DESIGN PROPERTIES TO QUALITY ATTRIBUTES:

Based on the review information, the design properties “abstraction” has an effect on the following quality attributes like: flexibility, extendibility, functionality and effectiveness. Flexibility, understandability and reusability have been influenced by encapsulation design property. Furthermore, coupling affects the understandability, reusability and extendibility. Low coupling is considered good for understanding the design, extendibility, and reusability while the higher coupling influences adversely these quality attributes.

The communication of the objects is the message passing, which directly affects the functionality, effectiveness and reusability. The composition can increase flexibility, reusability, effectiveness, extendibility and functionality. Furthermore, the use of inheritance influences reusability, effectiveness and extendibility. While, it has adversely effects on the understandability and the flexibility. The use of polymorphism affects to increase the flexibility, functionality and effectiveness. It also makes the design difficult to understand. The design property “complex” affects the design reusability and flexibility.

Table 6: the relationship between the quality attributes and design property

Software design quality attribute	Reusability	Extendibility	Flexibility	Functionality	Correctness	Effectiveness	Understandability
Design size	High						
	Low						
Hierarchies							
Abstraction		High	High	High		High	
		Low	Low	Low		Low	
Encapsulation	High		High				**High**
	Low		Low				**Low**
Cohesion	High						**High**
	Low						**Low**
Coupling	Low	Low					**Low**
	High	High					**High**
Composition	High		High	High			
	Low		Low	Low			
Inheritance	High	High	Low	High		High	**Low**
	Low	Low	High	Low		Low	**High**
polymorphism		High	High	High		High	**Low**
		Low	Low	Low		Low	**High**
Messaging				High		High	
				Low		Low	
Complexity	Low		Low				**Low**
	High		High				**High**

4 PROPOSED FRAMEWORK:

The proposed framework to evaluate the object oriented design is fitted to different stages as shown in the following diagram:

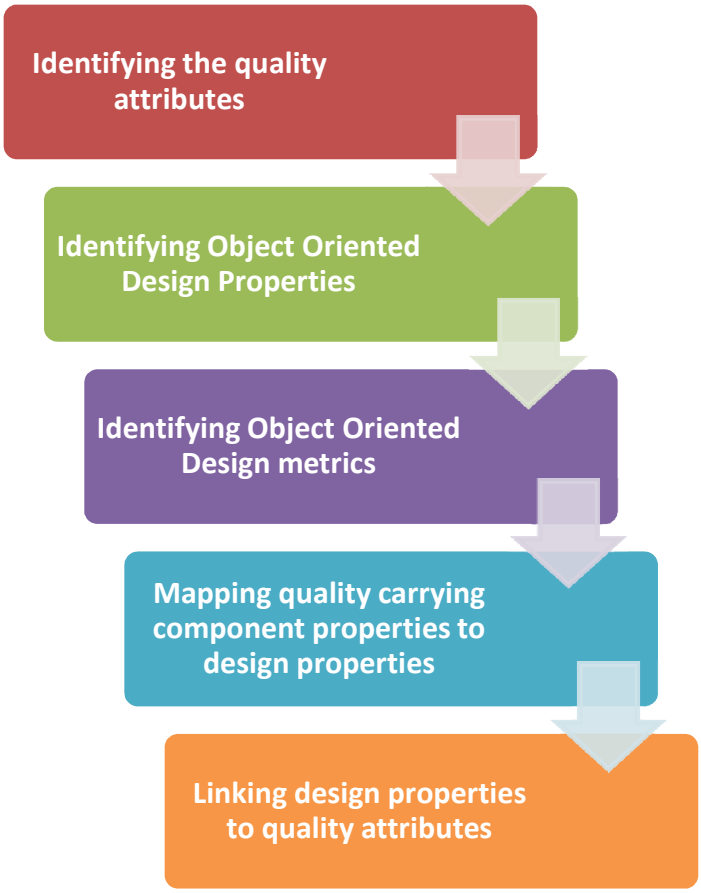

Figure 2 : The proposed framework

5 CONCLUSION AND FUTURE RESEARCH:

In this paper, the framework to evaluate the quality of object oriented design has been developed as a hybrid of previous quality models. My future work is to validate the proposed framework by applying the design metrics on real UML design diagram and study the effects of the object oriented design properties on the quality attribute.

REFERENCES

[1] Bansiya, Jagdish, Davis, Carl G., 2002. A hierarchical Model for Object Oriented Design Quality Assesment.
[2] Chauhan, Ritu et.all . 2014. Estimation of software quality using object oriented design metrics.
[3] Jamali, Sayyed Mohsen. 2006. Object Oriented Metrics (A survey approach)
[4] Pressman, Roger S. Software Engineering a paractioner's approach McGraw- Hill series in computer science,2001.