J. Basic. Appl. Sci. Res., 3(1s)539-544, 2013

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

An Approach for Quality Control Management in Object Oriented Projects Development

Mansoure Ashourion[1]*

Faculty Member of Payamenoor University-Computer Engineering- Iran- Esfahan

Received: June 10 2013
Accepted: July 8 2013

ABSTRACT

From the beginning of software creation, defining its quality has been discussed as a basic challenge. Among other things, object-oriented appearance also presents a vital need for drawing up metric in accordance with concepts of approach. In addition, lack of a well-defined and flexible method to draw low level quality characteristics to high level metrics for software is also a problem that no one has found a good answer for it so far. In this study, we have tried to draw low level quality metrics to high level one for object oriented design by defining a well-defined and flexible method.
KEYWORDS: quality, object oriented design, quality metrics, quality model.

1. INTRODUCTION

Many metrics and quality models for analyzing object-oriented soft wares can be used just after completing software and/ or when it is near to. They work on the basis of information obtained by product implementation , so conclusion would be defined so later than what ever could help to improve characteristics of software quality. Indeed, during the first processes of development (requirements and design) we need metrics and models to guarantee that analysis and design have desirable characteristic that could led to the final product of a good quality. These help developers to solve problems, remove unsuitable cases to standards and also to eliminate inappropriate complexities during those first processes of the cycle of production. It should much help to design effective tests, better management of project and resources and the most important to reduce future tasks during implementation and after that.

Now, for finding a way that could measure quality of object-oriented software, below problems should be solved. First of all, we need a method which can recognize object-oriented software quality faster than the possible time in cycle development. Indeed, we should adjust or invent a metric which:

-First could be runnable during that first processes and

-Second showing certain results and estimations as well.

We should also find a measurable way to relate object-oriented measurable characteristics to high level desired attributes of software. This relation needs to be defined and also various quality models should be provided so that by them we can be able to modeling various software environments and can answer various theories about theirs quantities. Also process of measuring should be happened automatically as well and they should be changed into tools for automatic derivation data from large software system that could be adjusted easily when it is necessary.

2- Previous quality models

One of the first qualitative models for software products was suggested by Mccall and his colleagues. Their model, the first model of this kind, knows software products quality as a hierarchy of factors, standards and metrics. International efforts also led to provide the standard of ISO 9126. However, all of the models have different compact in model definition, but they have a common problem.

They are all ambiguous to define details and low level metrics required for quality evaluation. Lack of clarity in these models guide developers insufficiently. The other problem of the first models is about their weakness to enumerate with relationship between quality attributes. Although, several high level attributes are used to describe product quality, but, generally, each model considers it as related only with subsets of attributes. Since effects of attributes on the all quality is not independent, so attributes or set of quality attributes have opposite effect on all the quality. For example flexibility attribute make accessing to low cost of maintenance difficult.

The other weakness of first models is about their inability to consider degree of effects for attributes. However, total quality of product is an outcome of set of all attributes, but each of them does not have an identical effect on product

This article is extracted from a plan that has been approved to support payamenoor university with subject" An approach for quality control management in object oriented projects development"

***Corresponding Author:** Faculty Member of Payamenoor University-Computer Engineering- Iran- Esfahan
E_mail : ashourion@pnu.ac.ir

qualities. Therefore the effect of an attribute requires a weight factor (a factor for measuring attributes). For instance in a large organization with complex network and real time process, probably performance and reliability have the most importance but in an organization with different infra structure, portability and extendibility are attributes which have the highest level of importance. It is difficult to determine set of quality attributes and it depends on management and business' objectives, competition, economy and duration time of the project.

Although the primary models provide high framework to start the projects, definition and evaluation of new metrics, communications and weights should be added to them.

3-Model development

For years, there was needed to have a framework for product-oriented quality models, so at first it was developed by Dromey. This framework removes some problems from the primary models such as Mccall and ISO 9126; also it defines a methodology for developing quality models in low to high phase that could guarantees recognition and calculating low level details. Dromey's quality model such as primary models supports analyzing high level quality attributes to visible properties of product items (requirement, design and implementation). There was three main elements in Dromey's general quality model:

-product properties which impress product quality,
-a set of high level quality attributes and
-their relationship.

Suggested methodology in this study, called "Quality Model for Object- Oriented Design" (QMOOD), extended Dromey's general quality model methodology.

In steps' definition, we will determine quality attributes and properties for design, metrics for object oriented design and object oriented design for items and specify changing low level attributes to high level in connections.

3-1-Determining quality attributes for design

QMOOD like ISO 9126 also considers attributes for functionality, reliability, maintainability and portability as the first set of quality attributes. Then each attributes was separately considered so that they can be evaluated from their effective in design quality and also having sufficient attributes.

Usability and reliability attributes removed because of being trended to quality of software implementation. Largely, the word of portability is more used for software implement so it replaced by extendibility which is more suitable for designing. Similarly, performance substituted with effectiveness. Finally, maintainability also replaced understandability. Thus, reusability is added as one of the most important attributes in object- oriented design qualities.

Flexibility of software systems is also one of the main characteristics either to develop or final user. We should determine the fundament of this attribute in design; also it should be supported by software architecture. Therefore, flexibility is added as a quality attribute as well.

Definition of quality attributes used for developing QMOOD model has been summarized in table 1.

Table 1: Definitions of Quality Characteristics

Quality Attributes	Definition
Reusability	This characteristic describes possibility of using designing for new issues without imposing important changes.
Flexibility	This characteristic describes possibility to change design. It is ability to design which caused to adopt software for providing related tasks.
Understandability	The design characteristic caused to easy understanding. It relates directly to the complexity of design structure.
Usage	The function which is allocated to class and will be given through class public interfaces.
Extendibility	It refers to presence and use of properties for current design that facilitates exerting of new requirements to design
Efficiency	This characteristic refers to the ability which is product and show desirable usage by concepts and object-oriented techniques

3-2. Determining properties of object-oriented design:

Design properties are clear concepts which can be specified by testing internal and external structure, communications and function of components, methods of attributes and classes. Evaluation of class definition by considering to their inherited relation with other classes and considering components attributes and their inner methods show main information about structural characteristics and class application and its objects.

Attributes of abstraction, capsulation, coupling, cohesion, complication and design size are also counted as structural characteristic and object-oriented as well. In contrast, messaging, aggregation, inheritance, polymorphism and Hierarchy of class have been represented by object-oriented design and then we only describe them in the design. As determined in table 2, the first version of QMOOD includes both collections.

Table 2. definition of Design characteristic

Design size	Number of classes used in Design
Hierarchies	It refers to different hierarchies are available in design. It is the number of unsprayed classes which have some children in design.
Abstraction	Hierarchy size of classes in design. Classes which have children in design present this characteristic.
Capsulation	Surrounding data and behavior has been defined in one structure. Particularly, it is applied to cases protect inside of object by defining "private" in object –oriented design.
Coupling	It describes dependence between an object to other design objects. It is the number of things that an object will use them to do these tasks.
Cohesion	It describes the relation between method and characteristic of object. A close similarity of parameters of methods with characteristics which describe cohesion is high.
aggrigation	Communicating size is defined by terms "component", "include" and "consisting of". Indeed, it describes the relation of gathering in object-oriented design.
Inheritance	The relation size between classes is as "is-a". This size relates to hierarchy level of classes.
Polymorphic	An ability which permit to change an object into another object with the same interface in runtime. Number of services defined for object dynamically when it is running.
Messaging	Number of general methods used by other classes as a service.
Complication	It describes a degree of difficulty in understanding an internal and external structure of classes and also their relationship.

3.Determining metrics for object-oriented design:

Each property is defined in QMOOD model refers to property or characteristic which can be made similar to one or more design metric. For object-oriented design, this information should include definition of classes and their hierarchies, declaration of methods with different kinds of their parameters and declaration of attributes.

By looking through current design metrics, we find that there are metrics could be used for defining some properties like abstraction, messaging and inheritance. In addition to these cases, some other design properties as capsulation and aggregation are unparalleled in object-oriented design metrics. However, necessary metrics, presented for recognizing complexity, coupling and cohesion; but these metrics can be calculated at the end of implementation, so we cannot use them for QMOOD model. This deficiency guides us to define five new metric. Data Access Metric (DAM), Direct Class Coupling Metric (DCC), Cohesion Among Methods of class (CAM), Measure Of Aggression (MOA) and Measure of Functional Abstraction (MFA) that they just need to design information for calculating. A complete set of metrics used in QMOOD is shown in table 3.

Table 3: Explanation of Design Metric

DSC	Design size in classes	This metric shows the total number of all classes available in design.
NOH	Number of hierarchies	This criteria shows hierarchy's number of design classes.
ANA	Average number of ancestors	It clarifies the average number of ancestors in one class. The way of calculating is in a way that number of classes in all branching paths is calculated by the origin of the structure of inheritance.
DAM	Data access metric	It shows the ratio of private characteristics to all ones in class. Its degree shows desirability in design (0<x<1). (The limitation 0&1)
DCC	Direct class coupling	It shows the number of classes that one class communicates with them directly. They include classes which are related to each other by attributes declaration and messaging.
CAM	Cohesion among methods of class	This metric calculates the relation between methods of class by using a list of parameters of methods. The metric is calculated by all common parameters of method with the maximum independence set of parameters of classes. Values near to (0&1) are desired. (0<x<1)
MOA	Measure of aggregation	It calculates the relation between part - whole by using characteristics' measures. This is performed by counting declarations which are the kinds of classes defined by the user.
MFA	Measure of functional abstraction	It shows the ratio of class inheritance methods to all methods used by number ones (the limitation is between 0&1→ 0<x<1)
NOP	Number of polymorphic methods	It is the number of methods can show Polymorphic behavior. These methods are displayed as virtual in C++.
CIS	Class interface size	It is the number of class public methods.
NOM	Number of methods	It is the number of all methods has been defined in class.

3-4. Determination of designing object oriented components:

Designing components that defines architecture of object oriented design include objects, classes and the relationship between them. Quality of class is defined by its constructors include attributes, methods and should other things are available in it, it will be clarified. The other component which one can define in object oriented design is a hierarchy of classes that organize their class family. Then, a set of components which could analyze, display and implement the object

oriented design, should have included attributes, methods, objects (classes), relations and the hierarchy of classes. Generally a special syntax is presented in all programming language object oriented in order to display these fundamental components. Since the object oriented programming language of C++ has been selected for displaying, design quality can be evaluated by identifying a similar case in these components. Automatic tools of QMOOD++ facilitate collecting metrics from components are designed in C++.

3-5. Drawing properties with quality attribute to design properties:

Properties include quality of designing components are classified on the basis of similar design properties. However, fundamental properties have high quality (attributes, methods and classes) but many of them are common. For example all of them have names, so name can be received as a property with quality attribute. When design has a high ability of understanding, name is named as descriptor, So complexity would be low. Capsulation is received for attributes, methods and classes as quality attribute that is directly drawn to similarity and other properties with quality attributes could also be drawn to a subset of eleven properties in table 2.

3-6. Allocating design metrics to design properties:

The purpose of this allocation is to find a good small set involved enough information for determining exact design properties. Design Size in Classes (DSC) and Number Of Hierarchies (NOH) are used for determination of two properties included size and hierarchies in QMOOD.

Abstraction refers the structure of design hierarchies and therefore is evaluated by average number of ancestors (ANA). The definition of capsulation property in design in table 2 refers to access of decelerating attributes which are described by DAM. Direct class coupling (DCC) is as a size for a direct relationship between two classes and therefore is used to evaluate coupling property in design. CAM, MOA and class interface size are used to evaluate cohesion property, aggregation and messaging in design. Inheritance is defined by subclass making of available classes; therefore it is clear that MFA could evaluate this design property well. For an object oriented design that displayed in C++, polymorphic property size in design is evaluated by number of virtual methods in class and determined by number of polymorphic methods (NOP). Chidamber used NOM as a characteristic of complexity and also Kamrer used it to define weighted method per class (WMC). When all method weight are considered the same, WMC and NOM provide the same evaluation. Design metrics used to evaluate 11 design properties are shown briefly in table 4.

Table 4. Metrics of designing for properties of design

Design Specifications	Separated design metrics
Design Size	Design Size in Classes (NOH)
Hierarchy	Number of Hierarchy (NOH)
Abstraction	Average Number of Ancestry (ANA)
Capsulation	Data Access Metric (DAM)
coupling	Direct Conjunct in Classes (DCC)
Cohesion	Connection Between Classes Method
aggregation	Collection size (MOA)
Inheritance	Size of Function Abstraction (MFA)
polymorphism	Number of Polymorphic (NOP)
Messaging	Classes Interface Size (CIS)
Complication	Number of Methods (NOM)

3-7. connecting designing properties to quality attributes

For expressing procedure of affecting design properties on designing quality and also connection between product specifications with qualitative attributes, complete study should be conducted in the field of developing object oriented procedures.

Information obtained in this study show that abstraction has a considerable effect on flexibility, product effectiveness, functionality and extendibility of product. Encapsulation increases flexibility, reusability and understandability of designing. Low coupling is a sign of understanding, extendibility and reusability, while high coupling has an opposite effect on the above mentioned attributes. Cohesion as well, reveals an important effect on understanding of design and reusability for product.

Relation of objects through message passing and as a result messaging is directly affecting functionality and product effectiveness and is an effective assistance to product reusability. However, inheritance increase reusability, functionality, extendibility and product internal effectiveness, it has a potential ability to impact negatively on flexibility and understand of design. Similarly, as precise aggregation of objects can considerably increase internal reusability, functionality and flexibility, its excessive and inaccurate uses well, mutually make designing understand a more difficult issue. Also, aggregation of objects influences effectiveness and extendibility of product. Polymorphisms as well acts as aggregation of

objects and increase product flexibility, extendibility, effectiveness and functionality and on the other side have a negative impact on design understanding, make its comprehension harder. It is clear that the direct meaning of complexity is obscurity of comprehending design. Generally, the more product complexity can case the more difficulty in its comprehension, and also it will have an undesirable effect on product flexibility and reusability.

Effect of each design properties on qualitative attributes have been shown at table 5. Sign on the top (↑) will show the positive effect of property and sign on down (↓) will show the negative effect on design metrics.

Table 5: Relation of designing properties and qualitative attributes

	Reusability	Flexibility	Comprehensibility
Design size	↑		
Hierarchy			
Abstraction			
Capsulation		↑	↑
Correction			
	↑		↑
aggregation		↑	
Inheritance			
Polymorphism		↑	
Messaging	↑		
Complexity			

3.7.1. Weighting connection between designing properties and quality attributes

Based on relations between designing properties and quality attributes presented in table 5, effect of each designing properties on quality attributes has been partially weighted, so that computed amounts to make same limitation for all of attributes. This area is selected between -1 to +1.

In first step of initialization it is consider +1and/or +0.5 values for positive effects and -1 and/or -0.5 values for negative effects.

After this initialization, result has been changed respectively until all computed values located in determine range. These result been shown in table 6. The reason of selecting this procedure is its simplicity.

Table 6- Computation formulas of quality attribute

Quality attribute	Computation formula
Reusability	Dependency × 0.25 + messaging × 0.5 + design's size × 0.5 – cohesion × 0.25
Flexibility	Capsulation × 0.25 + mixing × 0.5 + polymorphism × 0.5 – cohesion × 0.25
understandability	Capsulation × 0.33 + coupling × 0.33 – abstraction × 0.33 – cohesion × 0.33 - polymorphism × 0.33 – complexity × 0.33 - design size × 0.33
functionality	Cohesion × 0.12 + polymorphism × 0.22 + messaging × 0.22 - design size × 0.22 + Hierarchy × 0.22
Extendibility	Abstraction × 0.5 + inheritance × 0.5 + polymorphism × 0.5 – cohesion × 0.5
Efficiency	Abstraction × 0.2 + capsulation × 0.2 + aggregation × 0.2 + Hierarchy × 0.2 + polymorphism × 0.2

3-8. Correction and adaptation of model

Changes in qualify model (QMOOD) has been simply occurred and so we can implement various scales, point of view and different goals. In lowest level, evaluating metrics of design characteristic has been changed or a different set of characteristic design has been selected in order to evaluation qualify or evaluative quality attributes has been converted. Effective relations of design properties on quality attributes and their scales can change upto the best view of organization's goals will display.

4. Results and future researches

Hierarchy model (QMOOD) was developed to order evaluation of high level quality attributes of object oriented design. Also ability of model to realization of whole qualification of software according to the design information, evaluating model about several project with same requiring and its comparison with the result of human's evaluation was confirmed. Tool QMOOD++ is used for summarizing the design metrics' information and calculating simple qualification attributes and well definition.

Beside simplicity of using and clearness of assumption, result of model and human evaluation are so close to gather. This can clear the efficiency of these kinds of models to detect software product qualification. Beside problem of detection software qualification, QMOOD model will present tool for various models and approaches for rapidity evaluation of real projects which lead to close and join different ideas.

1. REFERENCES

2. R. Geoff Dromey, A Model for Software Product Quality, IEEE Transactions on Software Engineering, v.21 n.2, p.146-162, February 2005[doi>10.1109/32.345830]
3. Dromey, R.G. 1998. Software product quality: Theory, model, practice, see publications at http://www.sqi.gu. edu.au.
4. S. R. Chidamber and C. F. Kemerer: A metrics suite for object-oriented design، *IEEE Trans. on Software Eng.*، Vol. 20، No.6،4 pp.476-493 (2007).
5. Alistair Cockburn. *Writing Effective Use Cases.* Addison-Wesley، 2001.
6. L. Fischman، K. McRitchie. “A Size Metric For UML-Derived Software”
7. Bunge، M.، Treatise on Basic Philosophy : Ontology I : The Furniture of the World، Boston: Riedel، 2009.
8. Kemerer، C.F.، "Reliability of Function Points Measurement: A Field Experiment"، *Communications of the ACM*، vol. 36، pp. 85-97، 2004
9. Vessey، I. and Weber، R.، "Research on Structured Programming: An Empiricist's Evaluation"، IEEE Transactions on Software Engineering، vol. SE-10، pp. 394-407، 2010.