



Product vs. Design Quality of Object-Oriented Software

Vibhash Yadav¹, Professor Raghuraj Singh²

¹Computer Science & Engineering Department
Krishna Girls Engineering College
Kanpur, INDIA

²Computer Science & Engineering Department
Harcourt Butler Technological Institute
Kanpur, INDIA

Abstract: Assessment of quality of object oriented software during design phase has been a prime objective among researchers in software engineering discipline. There are many available tools that offer the software product quality from its code. Unfortunately, assessment of design quality has been more of a theoretical issue and the software engineering market lacks readily available tools for this purpose.

We make use of a set of UML diagrams created during the design phase of the development process to calculate design quality. UMLet software has been used for creating the UML diagrams. Design metrics are fetched from the UML diagrams using a parser developed by us and design quality is assessed using a hierarchical model of software design quality.

To validate the design quality, we compute the product quality from software code that corresponds to the UML design diagrams using available tools METRIC 1.3.6, JHAWK and Team In a Box. The objective is to establish a correspondence between design quality and product quality of object oriented software and thus identifying the design metrics that play decisive role in the quality of a software product. For this purpose, we have chosen seven software of known quality as low, medium and high.

A strong correlation has been observed among the results obtained from the two approaches i.e., between design quality and the product quality, which validates our chosen metrics set for determining software design quality.

Keywords: Object Oriented Software, DQL, PQL, UML, Design Quality Metrics

I. Introduction

Today, many metrics and quality models are available for assessment of quality of object oriented software. Most of these can only be applied when the product is completed or nearly completed. They primarily make use of the software code to fetch the software metrics used in determining its quality. This makes it difficult to restructure the design for its improvement which is possible only at the early stage of software development process. Detection of a faulty design at the later stages of software development bears a heavy price in terms of effort and cost. Thus, there is a need for models which could make an assessment of the software quality during the design phase of software development [1]. In this context, we consider the following issues in the foregoing sections:

- Analysis of existing measures and metrics for design strength and quality of object oriented software.
- Development of a design quality model based on UML diagrams.
- Design and development of a Parser which extracts the chosen metrics values from UML diagrams.
- Evaluation and verification of the proposed design quality model using existing product quality models.

Taking a different set of examples, few concepts of this work were described by us elsewhere during initial stages [2] [3]. Here, in this paper we present the complete work.

II. Related Work

Various efforts have been previously made to measure the design strength and design quality of object oriented software systems. Till date many algorithms/models have been proposed for the improvement of design & quality.

Measurement of software quality is based on

1. Complexity
2. Design Quality Metrics

The research contributions are aimed at providing operating definitions & metrics of software complexity. Design quality has become of increasing importance to cope with the continuously growing size of software systems. Research started to distinguish between the complexity due to poor design quality & the inherent complexity of the software due to requirements. With the advent of object oriented programming coupling,

cohesion, inheritance & abstraction have been identified as the basic properties of software design quality. Based on above properties, number of metrics have been proposed to evaluate the design quality of object oriented software.

Based on his experience in object oriented software development, Lorenz (1993) proposed eleven metrics known as Object Oriented Design (OOD) Metrics [4]. In 1994, Chidamber and Kemerer [5] proposed six OOD and complexity metrics, which later became the commonly referred to CK metrics suite.

Usefulness of the CK metrics suite for predicting the probability of detecting faulty classes, Basili and Colleagues (1996) performed an empirical study over the CK metrics [6]. Bansia J. et. al. [1] presented a hierarchical model for assessment of design quality of object oriented software systems in quantitative terms using various lower level and higher level quality metrics / parameters.

The MOOD metrics set refers to a basic structural mechanism of the OO paradigm as encapsulation (MHF and AHF), polymorphism (PF), message-passing (CF) and are expressed as quotients [7], [9], [10]. The QMOOD is a comprehensive quality model that establishes a clearly defined and empirically validated model to assess OOD quality attributes such as understandability and reusability and relates it through mathematical formulas, with structural OOD properties such as encapsulation and coupling. The QMOOD model consists of six equations that establish relationship between six OOD quality attributes and eleven design properties. All these are measurable directly from class diagrams, and applicable to UML class diagrams.

III. Problem Formulation and Proposed Methodology

In order to assess the design quality of object oriented software, following tasks have been identified:

- Obtaining UML design diagrams as well as code for the object oriented software.
- Identification of optimal set of object oriented software design measures/metrics.
- Extraction of Information needed to calculate each metric from the design specifications in various forms such as XML specifications etc.
- Presentation of Information on each quality metric in appropriate format for easier interpretation of the metric by the user.
- Development of a metric repository for various projects for comparison

We make use of two approaches for the assessment of software quality. The first approach takes UML diagrams of object oriented software as input and calculates design quality. In the second approach, code base is taken for the assessment of the product quality. The results of both the approaches are then compared. Regression and other analysis approaches have been used to show the relationship between design quality and product quality and to establish the co-relation between the two metrics. Design quality and product quality are assessed for three category of software. The categories include software with well established and known quality i.e. good, bad, and medium. Choosing one or two benchmark software of each category makes the comparison of results easier and thus, forms a basis for establishment of relationship between software design quality and product quality.

IV. Determination of Design Quality

A. UML Diagrams as Program Input

The approach discussed in this section deals with the assessment of design quality of software designed on the object-oriented philosophy. The first decision that must be taken before going any further concerns the notation to be used to represent the design. The Unified Modeling Language (UML) unifies the industry's best engineering practices for modeling systems and allows the designer to specify a system by making use of nine different types of diagrams [11]. In this approach, we use a set of UML diagrams used to represent design of the object oriented software. We analyze a subset of these diagrams depending on the metrics that are used to evaluate the system.

Various important issues related to this approach are:

- Selecting the suitable UML diagrams for extraction of values for the identified metrics.
- Defining the methodology to retrieve the metric values from the UML diagrams.
- Selecting the standard design quality attributes and properties.
- Formulation of relationship / correspondence between quality attributes and quality properties.
- Defining design quality attributes and establishing their relationship with the design quality.
- Assessment of the design quality of software designed on object oriented philosophy.

For rapidly drawing the UML diagrams with a pop-up-free and light weight user interface, we have used Open source Java tool named UMLet version 9.03 under GNU (Generic Public License) [12]. It supports editing of all UML diagrams in .uxf format and thus provides a solid base to which quality measurement capabilities can be added. System design in the form of UML diagrams is given as input to the UMLet tool. Figure 1 shows

a screenshot of UML diagram on UMLet. Following five types of UML diagrams have been selected for extraction of values for various identified metrics.

- Class diagram
- Object diagram
- Sequence diagram
- Activity diagram
- Collaboration diagrams

It has been established that all the metric related information can be obtained from the above mentioned five diagrams. UMLet creates .uxf document along with the UML diagrams and from .uxf we get the XML document. This XML document is used to extract information for calculating quality metrics/measures. The metric related data obtained through UMLet, is saved in XML format which is the universal format for structured documents and data on the web.

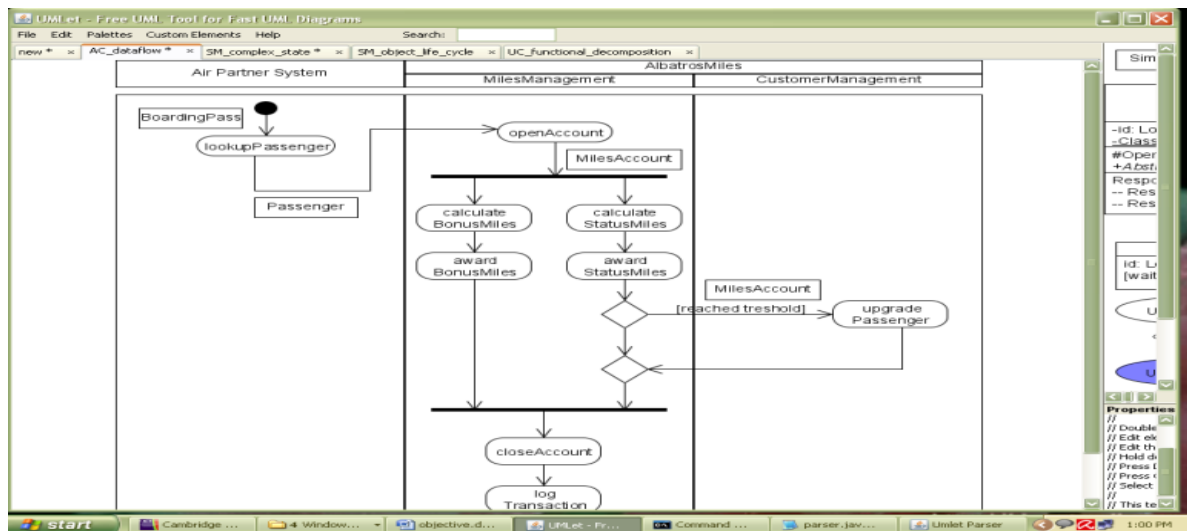


Figure 1 A Screen Shot Of UML Diagram On UMLET

To process an XML document obtained from the UML diagrams, we use a parse that reads the file, confirms that the file has the correct format, breaks it up into the constituent elements which can be accessed for the assessment of quality. We used the Document Object Model (DOM) parser which reads an XML document into a tree structure. For processing of the XML document obtained from the UML diagrams, we used functionalities of DOM and SAX parsers by writing appropriate codes. Values of the quality metrics/measures have been obtained for the sample UML diagrams of selected projects. The DOM parser interface is standardized by the World Wide Web Consortium (W3C). The org.w3c.dom package contains the definitions of interface types such as document and element.

B. The Parser Algorithm

The Parser code has been written in Java. The Parser code along with snapshots is being provided as supplementary material. A simplified algorithm for the Parser designed and developed by us is given below:

[Displaying xml document as a tree]

- Open the file and load the document.
- Find the public private and protected methods from the construct used in UML diagram.
- Find the value of polymorphism by determining the total number of abstract members in all the classes divided by the total number of classes.
- Find the value of inheritance by dividing public members by the total number of members.
- Find the value of aggregation and composition by using construct of UML diagram.
- Find hierarchy by using the formula of finding length of the tree build.
- Find abstraction by adding the value of inheritance and aggregation composition.

[Computing cohesion]

- In order to calculate the class cohesion of class queue, we first measure the relatedness of each method with the identified groups.

- $R(M)$ = total parameters used in the method/ total no. of parameters in the class.
- Then compute cohesion of each class as Cohesion (X) = $\sum R(M)$ /total no of methods in class.
- Total cohesion = $\sum$ Cohesion (X)/total no. of classes in the system.

[Computing coupling]

- Data coupling is determined by modules that communicate by passing data.
- Stamp coupling is determined by modules that communicate by passing complete data structures.
- Control coupling is determined by modules that communicate by passing control information.
- Common coupling is determined by modules that communicate by sharing global data.
- Content coupling occurs when a module changes the data of another module.
- External coupling is the value of external devices that communicate for running the code smoothly.

C. Metrics for Design Quality Assessment

The Model used for design quality assessment is based on Hierarchical Model of Quality Measurement given in [1]. Attributes in higher order of hierarchy affect the overall performance however; it may be difficult to measure them. Thus, we have to move down in hierarchy to get lower order attributes which can be measured through a well defined set of metrics. Description of Eleven identified metrics for design quality is given in the table I.

Table I Design Quality Metrics

Sr. No.	Metric	Name of the Metric	Description
1.	DC	Design size in class	It counts the total number of classes
2.	NH	Number of hierarchies	It is a count of total number of class hierarchies
3.	ACA	Average count of ancestors	This metric count the average number of classes from which a class inherits the information
4.	DAM	Data access metrics	This metric is ratio of number of protected /private attributes to the total number of attributes defined in the class
5.	CC	Class coupling	It counts the value through which different classes are directly related to.
6.	CAM	Cohesion among methods in class	This compute the value through which methods of a class are related
7.	MA	Measures of aggregation	It count the number of data declarations whose types are user defined
8.	CPM	Count of polymorphic methods	This count the methods of polymorphic behavior
9.	CIS	Class interface size	It counts the number of public methods in a class
10.	TNM	Total number of methods	It counts the total number of methods in a class
11.	MOA	Measure of abstraction	It is the ratio of the number of the methods inherited by a class to the total number of methods accessible by member methods of the class

D. Design Properties related with Metrics

Description of Eleven (11) standard properties used for metrics is given in the table II.

Table II Design Properties

Sr. No.	Design Property	Description
1.	Design size	Number of classes used in a design
2.	Hierarchies	It is a count of number of non-inherited classes that have children in a design
3.	Abstraction	A measure of generalization aspect of a design.
4.	Encapsulation	Defined as the enclosing of data and behavior in a single construct
5.	Coupling	Interdependency of objects on one another

6.	Cohesion	Relatedness of methods and attributes in a class
7.	Composition	It measures “part of,” “consist of,” “has,” or “part whole” relationship.
8.	Polymorphism	Ability to substitute objects whose interfaces matches for one another at run time
9.	Messaging	Number of public methods available as services to other methods
10.	Complexity	Degree of difficulty in understanding and comprehending the internal and external structure of classes and their relationship
11.	Inheritance	Measure of is-a relationship between classes

E. Design Quality Attributes

Description of six standard design quality attributes is given in the table III.

Table III Design Quality Attributes

Sr. No.	Design Quality Attributes	Description
1.	Reusability	Reusability reflects the presence of object-oriented design characteristics that allow a design to be reapplied to a new problem without significant effort.
2.	Flexibility	Flexibility can be stated as the characteristic that allows the incorporation of changes in a design. It is the ability of a design to be adapted to provide functionality related capabilities.
3.	Understandability	Understandability is the property of the design that enables it to be easily learned and comprehended. This is directly related to the complexity of the design structure. Understandability is related to coupling.
4.	Functionality	Functionality is the responsibility assigned to the classes of a design, which are made available by the classes through their public interfaces.
5.	Extensibility	Extendibility refers to the presence and usage of properties in an existing design that allows for the incorporation of new requirements in the design.
6.	Effectiveness	Effectiveness refers to a design’s ability to achieve the desired functionality and behavior using object-oriented design concepts and techniques.

F. Formulation of Relationship between Design quality and Quality Attributes

Hierarchical Model for assessment of software design quality has been used for establishing relationship between quality metrics and design attributes and thus, finding the overall software quality [7][8].

We have categorized the product quality Index rank wise into High, Medium and Low based on the following threshold values:

$$\begin{aligned}
 \text{DQI} = \{ & \text{Low} \quad \text{if } 0 \leq \text{DQI} \leq 1.8 \\
 & \text{Medium} \quad \text{if } 1.9 \leq \text{DQI} \leq 2.49 \\
 & \text{High} \quad \text{if } 2.5 \leq \text{DQI} \leq 4 \}
 \end{aligned}$$

V. Product Quality Determination

The design quality assessed using the first approach as described in the previous section is validated by determining the quality of the corresponding code of the software using the tools *Metric 1.3.6*, *Team In a Box* both as plug-in for eclipse and *JHawk* [13][14][15]. We have selected seven (07) open source software with known design details and known quality as low, medium and high. For each of the software, we extracted a set of chosen software metrics. After applying various levels of normalizations on these metric values, we determined the product quality for the above software.

A. Selection of open source software

We chose these software systems on the basis of their design quality considering the designer of the software and the reputation in software market. The details of software selected for this purpose are given in the Table IV.

Table IV Chosen Open Source Software With Prior Known Design Quality

Sr. No.	Name of Software	Level of Quality (Known Priori)
1.	Page_Test	Low

2.	Person	Low
3.	Shopping Card	Low
4.	Doc	Medium
5.	ATM	Medium
6.	Vehicle	Medium
7.	Internet Banking	High

B. Selection of Metrics

The most important measures for software quality are the Cyclomatic Complexity, Lack of Cohesion of Methods, Weighted Method per Class and Lines of Code. A brief description of these measures which we used is given in the table V.

Table V Chosen Metrics For Design Quality

Sr. No.	Name of the Metric	Expected Value
1.	Cyclomatic Complexity (CC)	Low value required
2.	Lack of Cohesion of Method (LCOM)	Low value required
3.	Weighted Method per Class (WMC)	Low value required
4.	Lines of Codes of Methods (LOCM)	Low value required

C. Assessment of the Metric Values

To obtain the metric values for the chosen software, we used the tools Metric 1.3.6, Team in a Box and JHawk open source software. Using these tools we measured the metric values for all the seven chosen software. The un-normalized Values of metrics are given in the table VI. These metric values have been obtained by applying the evaluation tools on the chosen software.

Table VI Un-Normalized Values For Product Quality Metrics

Software	CC	LCOM	WMC	LOCM
Page_Test	1	0.476	1.020	22
Person	1	0.250	0.829	16
Shopping Cart	1	0.125	0.829	16
Doc	1	0.407	1.432	41
ATM	1	0.422	2.179	56
Vehicle	1	0.139	0.618	17
Internet Banking	1	0.156	0.400	45

D. Normalization of the Metric Values

We have performed normalization of metric values at three levels namely

- Primary normalization
- Normalization at comparison level
- Normalization at the quality rank level

In our work, firstly we have normalized the metric values to a standardized form using normalization model, so that they can be compared irrespective of software sizes and complexity. At primary normalization level metric values are scaled down to standard unit of 10 KLOC that provides a precise look to software systems disregarding their size. The primary normalization for each of the above mentioned software is calculated as follows:

If $M_1, M_2, M_3, \dots, M_n$ are the individual metric values then Primary Metric Normalization value is given by

$$N_i = (M_i * 10k) / LOCM \quad i = 1, 2, 3 \dots n \quad (1)$$

The results after primary normalization are summarized in Table VII.

Table VII Normalized Values For Product Quality Metrics

Software	CC	LCOM	WMC
Page_Test	454.54	216.36	463.63
Doc	243.90	99.26	349.26
Person	625	156.25	518.12
ATM	178.57	75.35	389.107
Vehicle	588.23	81.76	363.52
Internet Banking	222.22	34.66	88.88
ShoppingCart	625	78.12	518.12

Comparison level normalization is performed by setting a metric threshold value among the comparing software group and by calculating the percentile metric among peers [18]. Primary Normalized Metric, N_i is set to percent for the comparing software, S_j it holds the maximum value and for the other software systems $S_1, S_2, S_3, \dots, S_k$. N_i value is calculated to percentile normalization, PN_{ik} such as:

$$PN_{ik} = (N_{ik} * N_{ij}) / PM_{\max} \quad (2)$$

If N_i is max for S_j then N_{ij} (primary normalized metric, N_i for S_j) is named as Maximum percentile normalized metric and changed as $PN_{\max} = N_{ij}$ and another change done as $N_{ij} = 100$. The resultant metrics after comparison level normalization are summarized in Table VIII.

Table VIII Normalization At Secondary (Comparison) Level

Software	CC	LCOM	WMC
Page_Test	72.72	100	89.48
Doc	39.024	45.87	67.40
Person	100	72.21	100
ATM	28.57	34.82	75.09
Vehicle	94.11	37.78	70.16
Internet Banking	35.55	16.01	17.15
Shopping Cart	100	36.10	100

Finally, the design quality ranking level normalization is done with weighted normalization average calculation, defining range and providing design quality rank to each software systems to indicate their overall product quality level.

If $PN_1, PN_2, PN_3, \dots, PN_n$ are individual percentile normalized metric values of a particular software S_j , then weighted normalization average for S_j is

$$W(NA_j) = (\sim PN_i) / n \quad (3)$$

Range defines metric threshold value for weighted normalization average for any particular software and lies between zero and percent inclusive such as:

$$0 \leq w(NA) \leq 100$$

Software System Normalized Quality Rank is Product Quality Level (PQI) used for the comparing software systems such that

$$PQI = \begin{cases} \text{High} & \text{if } 0 \leq w(NA) \leq 25 \\ \text{Medium} & \text{if } 25 < w(NA) \leq 75 \\ \text{Low} & \text{if } 75 < w(NA) \leq 100 \end{cases}$$

Normalized metrics at quality rank level are summarized in Table IX.

Table IX Normalization At Quality Rank Level

Software	PQI	Calculated PQI	Actual PQI
Page_Test	87.4	Low	Low
Person	90.73	Low	Low
Shopping Card	78.7	Low	Low
Doc	50.76	Medium	Medium
ATM	46.16	Medium	Medium
Vehicle	67.55	Medium	Medium
Internet Banking	22.90	High	High

VI. Comparison And Results

In this work, we have taken two approaches. The first is design based approach and the other is code based approach. The methodologies of both are compared below:

In design based approach basically the quality of software, represented through Design Quality Index (DQI) has been calculated using the UML diagrams. The UML diagrams are converted into XML document which are parsed using DOM parser to obtain the final values.

On the other hand in code based approach the entire software is coded first and then it is parsed to obtain the final metric values.

Table X summarizes the results obtained from the two approaches. Both approaches correctly place chosen software at their known gradation category. This asserts the prominent role of the chosen design metrics CC, LCOM, WMC and LOCM (as specified in Table V) in assessment of the design quality at early phases of software development.

Table X Design Quality And Product Quality Obtained From The Two Approaches

Software	PQI (From Software Code)	DQI (From UML Diagrams)	Quality
Shopping Card	90.73	1.12	Low
Page_Test	87.4	1.33	Low
Person	78.7	1.218	Low
Vehicle	67.55	1.967	Medium
Doc	50.76	2.058	Medium
ATM	46.16	2.12	Medium
Internet Banking	22.90	3.606	High

VII. Conclusions

In this paper we have done comparative assessment of design and product quality of the software falling in the categories of low, average and high quality and have established the correspondence between the two approaches. This provides a strong base for the effectiveness of the proposed approach for metric based design quality measurement of object-oriented software using UML design diagrams. Using this approach, we have been able to identify the Metrics set that play decisive role in both design quality as well as product quality.

VIII. References

- [1] Jagdish Bansiya and Carl G Devis: A Hierarchical Model for Object Oriented Design Quality Assessment, IEEE transactions of Software Engineering, Vol 28, No. 1, January 2002.
- [2] Vibhash Yadav and Raghuraj Singh, Validating Object Oriented Design Quality using Software Metrics, at International Conference on Advances in Electronics, Electrical and Computer Science Engineering - EEC 2012. Proceedings published in International Journal of Computer Science and its Applications (IJCSIA), December 2012 Issue (Volume 2 Issue 3).
- [3] Vibhash Yadav and Raghuraj Singh, Predicting Design Quality of Object-Oriented Software using UML Diagrams at 3rd IEEE International Advance Computing Conference (IACC-2013), February, 2013.
- [4] Lorenz, M. Object Oriented Software Development: A Practical Guide, Englewood Cliffs, N.J., Prentice Hall, 1993.
- [5] Chidamber, S. R., and C. F. Kemerer, A Metric Suite For Object Oriented Design, IEEE Transactions Of Software Engineering, Vol. 20, 1994. Pp. 476-493.
- [6] Victor R. Basili, Lionel C. Briand, and Walcélio L. Melo. A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, 22(10):751–761, 1996.
- [7] S. R. Chidamber, C.F. Kemerer, "A metrics suit for object oriented design", IEEE Trans. On Software eng., vol. 20, no. 6, p.p. 476-493, June 1994.
- [8] C. Otieno, W. Mwanji and S. Kimani. Framework to access Software Quality in ERP System in 7th JKUAT Scientific Conference 2012
- [9] Chidamber, S. R., and C. F. Kemerer, A Metric Suite For Object Oriented Design, IEEE Transactions Of Software Engineering, Vol. 20, 1994. Pp. 476-493.
- [10] Ul Kulkarni, YP Kalshetty and Vrushali G Arde: Validation of CK metrics for Object Oriented Design Measurement, Third International Conference on Emerging Trends in Engineering and Technology, ICETET. 2010.
- [11] Booch G., Rumbaugh, J. and Jacobson, I., The Unified Modeling Language User Guide, Addison Wesley; ACM Press, 1999.
- [12] UMLET [www.umlet.com] access date December 1012.
- [13] [Metric 1.3.6] Web Resource at www.metrics.sourceforge.net/ access date May 2013
- [14] [Team In a Box] Web Resource at www.team341.com/tiab.html access date May 2013
- [15] [JHAWK] Web Resource at www.virtualmachinery.com/jhawkprod.htm access date May 2013.