A Proposed Model to Determine Software Quality Specifically for Object Oriented Concept.

Vishal Bansal\textsuperscript{1} & Pooja Ahlawat\textsuperscript{2}

\textsuperscript{1}M.Tech Student, Department of CSE, R. N. College of Engineering & Management
\textsuperscript{2}Assistant Professor, Department of CSE, R. N. College of Engineering & Management

\textbf{Abstract:} The main reason behind failure of lots of software is poor quality thus estimation software quality become an important task in software industry. By late estimation of software quality results in ineffectiveness, late delivery and most important poor quality of software product. For this, an early estimation towards pre-released software quality plays an important role in shorting the time and by increases probability of project success. Metrics play an important role by deciding the usage pattern of resource of the industry as they are very valuable to the industry. This paper represents various metrics for estimation software product.

\textbf{Keywords:} MOOD Metrics, Ck Metrics, QMOOD Metrics etc.

1. Introduction
To measure the quality of software product in terms of durability, performance and reliability some metrics are required. Thus metrics provides us by a way to measure the quality of work done on product during development in relation of cost and time consumed. Object oriented software metrics directly focuses on the issues like complexity, reliability and robustness of the software developed using object oriented design methodologies. While the software in its development stage, it is desirable that the complexity levels at every stage should be minimized to make the end product more reliable and manageable. Object oriented metrics provides all parameters through which one can estimate the complexities and quality related issues of any software at their early stages of development [1].

2. Literature Review
Over the past years, with the invent of new methodologies and techniques, many process driven management approaches have been developed to address the problem of detecting and correcting design flaws in an Object Oriented software system using metrics. Moreover, with the ever increasing number of software metrics being introduced the project managers find it hard to interpret and understand the metric scores. Chidamber and Kemerer are the predominantly referenced researchers, they proposed 6 metrics- Weighted Methods per Class (WMC), Response sets for Class (RFC), Lack of Cohesion in methods (LCOM), Coupling Between Object Classes (CBO), Depth of Inheritance Tree (DIT), Number of Children of a class (NOC), with the help of which various software quality attributes (e.g. efficiency, complexity, understandability, reusability, maintainability and testability) can be measured. They claim that using several of their metrics collectively can help project managers and designers make better design decisions [2]. MOOD metric set model, proposed by Abreu [3] is another basic structural method of the object-oriented paradigm. They were defined to measure the use of object-oriented design methods such as inheritance ( MIF (Method Inheritance Factor), AIF (Attribute Inheritance Factor) ) metrics, information hiding (MHF (Method Hiding Factor), AHF (Attribute Hiding Factor)) metrics, and polymorphism PF (Polymorphism Factor) metrics. Abreu firmly suggested that metrics definitions and dimensions should be justified as they play important role in designing the object oriented metrics. Maintainability Estimation Model for Object - Oriented software in design phase (MEMOOD), estimation the maintainability of UML class diagram in term of understandability and modifiability and developed a multivariate linear model [4]. Object – Oriented process are used as a solution to software development problems. Object – Oriented development use to reduce the maintenance effort that not based on reliable experimentation [5]. The Halstead complexity is used for measuring maintainability. It shows the results that confirmed partially our assumptions that need to be evaluated with future uses [6]. The types of models are used that give us a vocabulary and a tool that allow us to discuss how to maintain software so as not to make it deteriorate. Verifying and valid verification measurements are used [7]. Study on the empirical evidence using some object –oriented metrics that can effectively predict maintainability of software systems. These metrics are such as size, inheritance, cohesion and coupling [8]. It presented a concern-oriented framework which supports the instantiation and comparison of concern measures. In this paper there is a rich body.
of ideas regarding the way to address concern measurement [9]. When more and more attentions are focused on the quality of the software, it’s reasonable to believe that the software complexity metrics will be sit on its right place that is the main purpose of a survey on metrics of software complexity [10]. Measure the software metrics and Reliability that try to define how software is reliable and easy to maintain, which free from errors, faults and failure [11].

3. Software Quality Parameters
From various sources I have found large no of parameters that affect the quality of a Software product. I have grouped some of them into two categories. One is software quality metrics and second is software quality factors.

3.1 Software Quality Metrics
Size
Size of a class is used to evaluate the sum of no of functions, dynamic object, and other functional points for better understanding of code by developers and for maintainers. Size can be measured by variety of ways. As the size of module increases defect density decreases thus increases the software quality.

Complexity
Define interrelationship of classes. In any software product interrelationship between different classes is high. Thus, to reduce the complexity we can refer to favor cohesion over co upling. Large module complexity greatly affects the quality of software.

Coupling
In 1974, Stevens et al. first defined the coupling in the context of structured development as “the measure of the strength of association establishment by a connection from one module to another.”

Comment percentage
Comment percentage can be calculated by total number of comments divided by total line of code excluding no of blank line in code. Higher comment percentage increase understandability and maintainability.

Effort estimation
Effort estimation can be calculated by dividing the size estimate (during a time period) by productivity (during the same period). More the effort estimation more will be the productivity.

Reusability
It takes advantages of the commonalities across different applications by reusing items from previous development into a new one. With the help of reusability, developers can create more complex and highly efficient product in lesser duration of time period. The cost that will incur in this process will be lesser than the cost that wills incur in developing a product from scratch.

3.2 Software Quality Factors
Correctness
It is the extent to which software product satisfies its specifications and fulfills all needs of customer. In the begging complete specifications are settled down with the customer regarding new product. The design of the system should meet all the specified requirements that all should be working correct. As the complexity of software product increases it become tougher to perform exhaustive testing on each and every possible path software can follow. For this a product must be tested on different platforms such as inspections and walkthroughs.

Portability
The effort required to transfer the program from one hardware/software environment to another system. We can say, transferring of product from one working environment into other one. This has many advantages as the lifetime of a good software product is more than that of that of hardware lifetime. Thus, while shifting between different hardware does not cause problem in software implementation.

Reliability
Measured by determining the period over which a software product can be remaining in execution without causing any crash. In real time systems such as air traffic controller or medical equipment’s reliability becomes the most important factor in the product quality.
**Maintainability**
The effort required to locate and fix an error in a software program. More the no of error more will be maintenance cost of the system and lower the efficiency. More often when new changes are made in a section of product then the complete documentation is required because the one who made changes will not be same person responsible for maintenance.

**Efficiency**
The capability of the software product to perform its operations related to the amount of resource used. It is important to the end customer in order to reduce running cost of the final software product. Efficiency is directly proportionally to the quality of product.

**Testability**
The effort required to test a program to ensure that it performs its intended function and the product is free from any kind of error. With lesser error found during product testing efficiency of product will increase.

**Information hiding**
It is a way of designing routines such that only a subset of the module’s properties, its public interface, is known to users of the module. This hides the information from user that is of no use of them and hence improves security of the data in the product.

**Inheritance**
Defines component of base class can be used in its sub class. It decreases the complexity by reducing the number of operation’s and operators, but this abstraction of objects can make maintenance and design difficult.

**Integrity**
Defines the extent to which each module of a software product can be interlinked with each other.

4. Proposed Model
A new proposed model can be defined to calculate the quality of the object oriented software product on the basis of factors and metric defined earlier.

![Figure 1: Basic Structure of Proposed Model](image-url)

Basic unit (level 0) values will be calculated from the inputs provided from the project directly. Output of which will be used as the inputs weights for level 1.

Metric level (level 1) values will be calculated by finding the relationship between input weights from level 0. And output will be act as input weights from Factor level (level 2). Factor level (level 2) can be calculated by input weights from level 1 along with the dependencies factor with them.
Basic structure of quality model calculates value of parameter at factor level (level 2) individually. This Extended proposed structure of quality model will combines the final result at level 2 to provide final output as quality (level 3) of final software product. Final value will be calculated on the basses of input provided from level 2 along with relationship between different factors.

5. Future Work
I’ve given a basic structure of proposed model for measuring quality of the software product. This will be a combination of some of the predefined metrics with new approach to find out relationship among them. I’m trying to have sufficient experimental results for finding out various dependencies between different metric forms thus finding out relationship among them. I’m working on completing implementation on this. An extended structure of this model can also be defined to find out relationship between different factors at level 2 in this model for calculating final quality of the product.

6. Conclusion
This paper represents a comparative study of different parameters which are responsible for quality of the object oriented software product in many terms and aspects. They are grouped into two major contagious namely factors and Metrics. In this paper I’ve given a base line of proposed model for measuring quality of the software product in two forms for finding out final results.

7. References


[5] Hall John M., the Maintainability of Object-Oriented Software.


