International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 12 (2014), pp. 1103-1112
© International Research Publications House
http://www. irphouse.com

A Proposed Model for Estimating Quality of Product Built Using Object Oriented Concept

Harish Mittal[1] and Aditya Jain[2]

[1]SPGOI College of Engineering, MD University, Rohtak, Haryana
[2]M.Tech Student, SPGOI College of Engineering, MD University, Rohtak, Haryana

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 proposed model for estimation quality of software product.

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. MOOD metric set model, proposed by Abreu [2] 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 [3]. 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 [4]. 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 [5]. 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. 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 [6]. 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 [4]. 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 [4]. 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 [4].

3. Proposed Quality Model

A new proposed model can be defined to calculate the quality of the object oriented software product.

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. Figure 1 Represents the basic structure of the quality model consists of three different levels.

Basic structure of quality model calculates value of parameter at factor level (level 2) individually. This complete 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. Figure 2 Represents the complete structure of this model consists of all four levels.

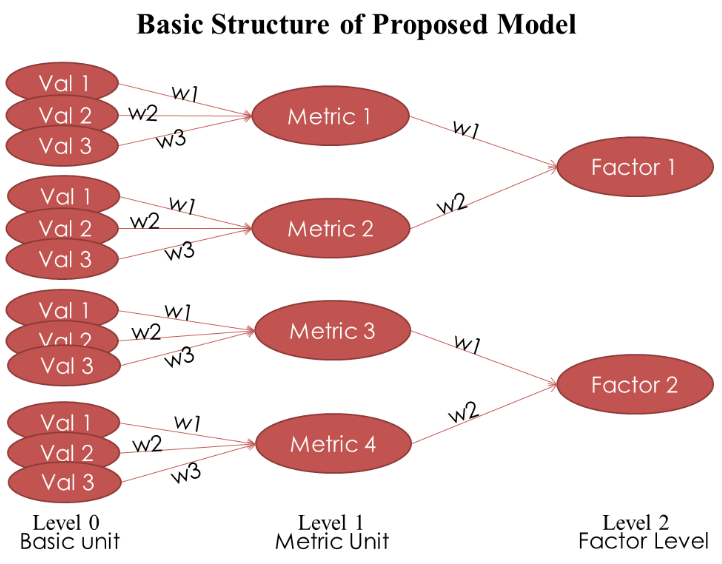


Figure 1: Basic Structure of Proposed Model [7]

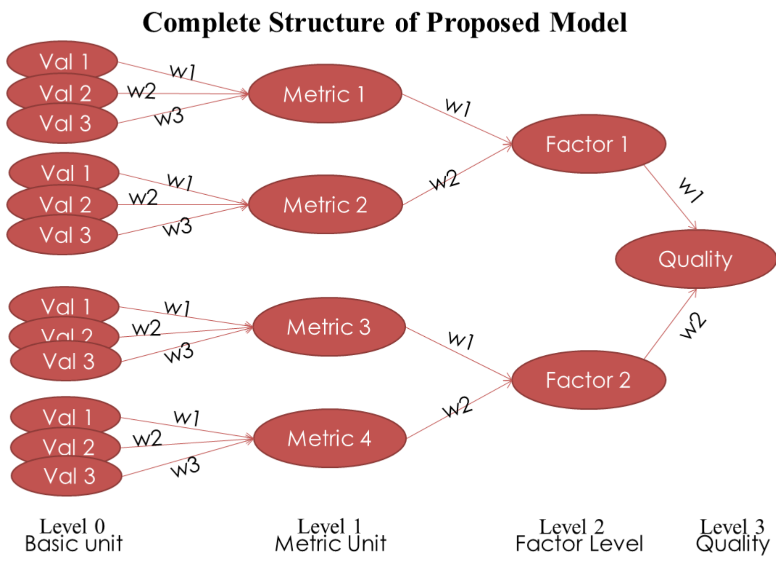


Figure 2: Complete Structure of Proposed Model [7]

4. Detail Description of Quality Model

This quality model calculates the quality of the product based on three factors Defect Density, Complexity and Change Effort. These factors depend upon various parameters such as Design Change, Error, Testability, Reliability, Cyclomatic Complexity, Information Flow, Comment Percentage, Correctness, Reusability, Portability and Modification. Some of these parameters are grouped into following metric values namely Change Effort, Efficiency, Corrective Change and Function Point. Figure 3 Represents overall Model description and Representation.

OverAll Model Description

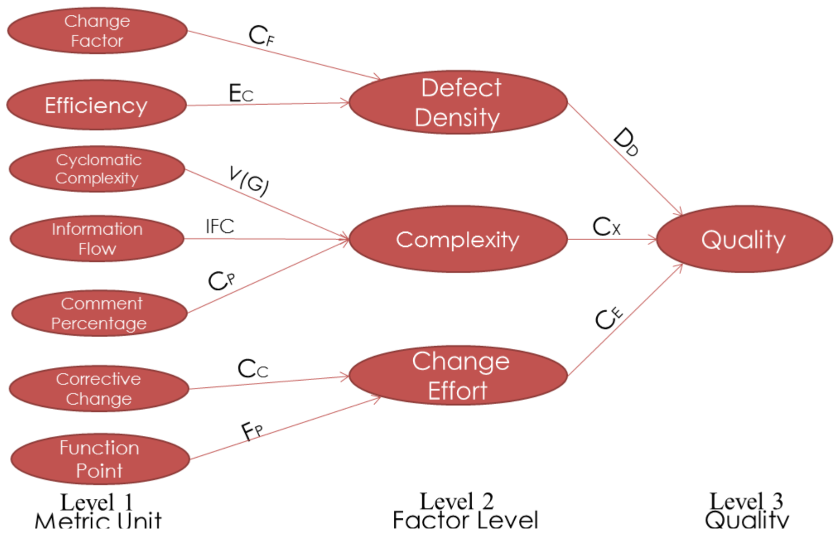


Figure 3: Overall Model Description

4.1 Detail Description of Defect Density Factor

Detail description of Defect Density is illustrated in the Figure 4. Defect Density factor will be depended upon Change Factor and Efficiency further Change Factor will be depended upon Design Change and Error. And Efficiency will be depended upon Testability and Reliability. Table 1 will consist of all the Symbols defined for determining Defect Density. Design Change will be calculated on the bases of Changes made and the total changes required in the product. **DC** will be equivalent to **(CD / CL) * 100.** Error will be calculated on the basis of Error Corrected and the total Error left out in the product. **EE** will be equivalent to **(ER / EL) * 100.** Now, by this Change Factor **CF** can be determined by taking **Mean of DC & EE.** Testability will be calculated on the bases of Time spend in testing and total development time in the product. **TB** will be equivalent to **(TT / TD) * 100.** Reliability will be calculated on the basis of Mean time to failure and Total run time of the product. **RB** will be equivalent to **(TF / TR) * 100.** Now, by this Efficiency **EC** can be determined by taking **Mean of TB & RB.**

Defect Density **DD** factor will be calculated by taking the Mean of Change Factor Compliment **CF^C = 100 – CF** with Efficiency Compliment **EC^C = 100 – EC**. Thus Defect Density **DD = Mean of CF^C & EC^C.**

Detail Description of Defect Density

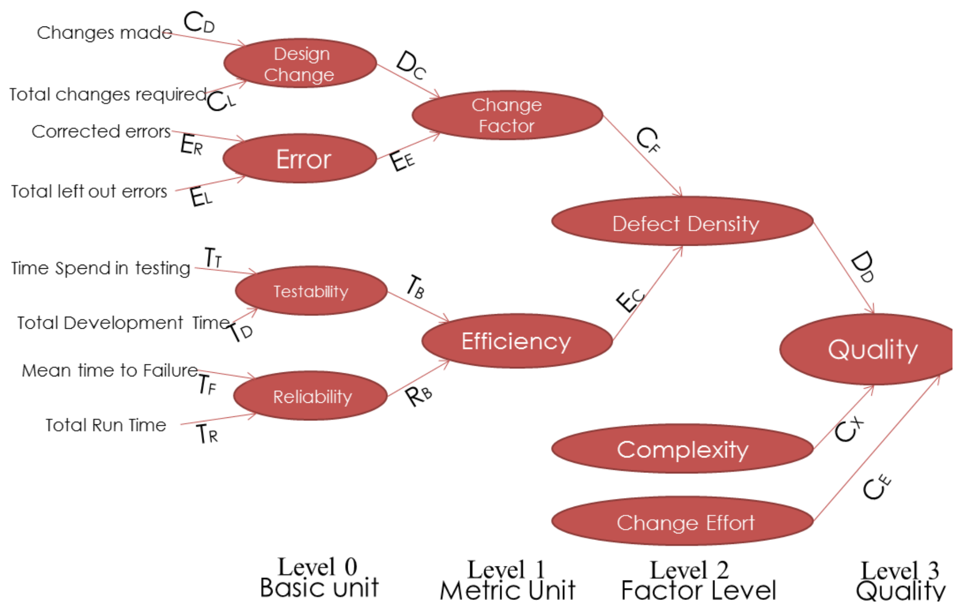
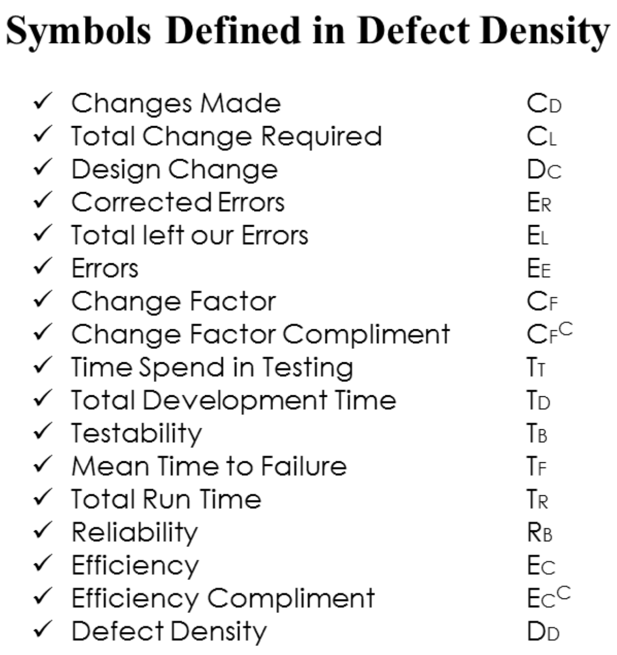


Figure 4: Detail Description of Defect Density

Table.1 Symbols Defined in Defect Density



Symbols Defined in Defect Density

✓ Changes Made	C_D
✓ Total Change Required	C_L
✓ Design Change	D_C
✓ Corrected Errors	E_R
✓ Total left our Errors	E_L
✓ Errors	E_E
✓ Change Factor	C_F
✓ Change Factor Compliment	$C_F{}^C$
✓ Time Spend in Testing	T_T
✓ Total Development Time	T_D
✓ Testability	T_B
✓ Mean Time to Failure	T_F
✓ Total Run Time	T_R
✓ Reliability	R_B
✓ Efficiency	E_C
✓ Efficiency Compliment	$E_C{}^C$
✓ Defect Density	D_D

4.2 Detail Description of Complexity Factor

Detail description of Complexity is illustrated in the Figure 5. Complexity factor will be depended upon Cyclomatic Complexity, Information Flow and Comment Percentage. Table 2 will consist of all the Symbols used for determining Complexity. Cyclomatic Complexity is defined by **Thomas McCabe** in 1976 and it is based upon no of Edges, Vertices and No. of connected components in a graph.

Detail Description of Complexity

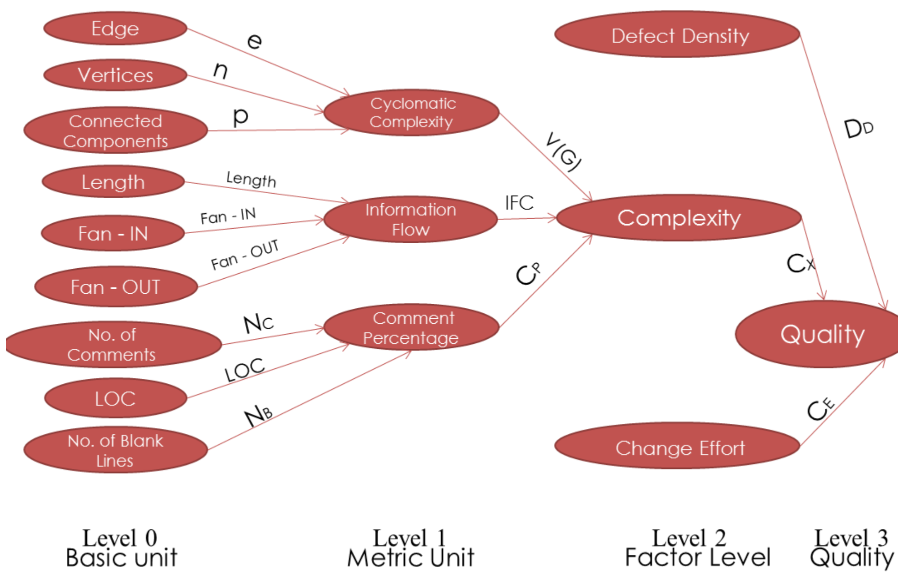
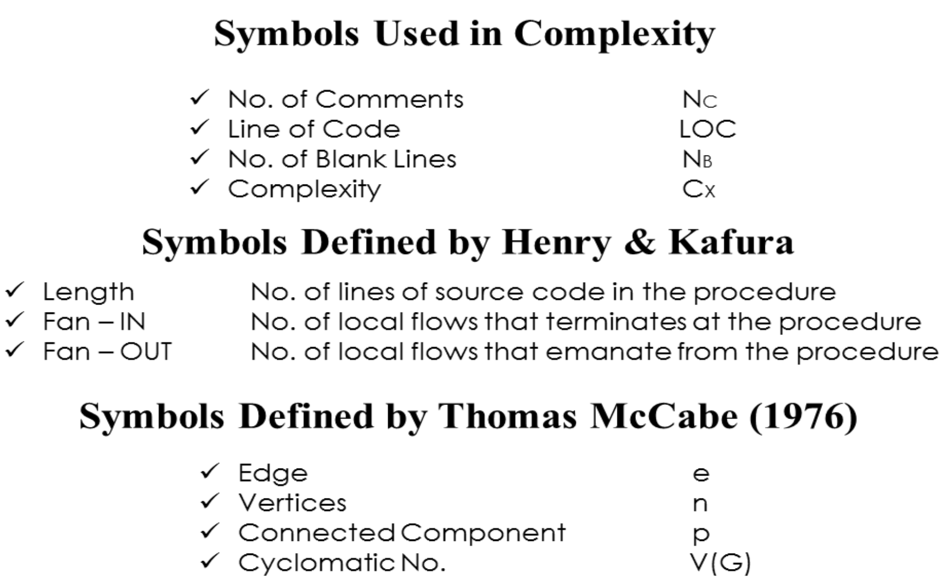


Figure 5: Detail Description of Complexity

Table.2 Symbols Used in Complexity



Symbols Used in Complexity

✓ No. of Comments	Nc
✓ Line of Code	LOC
✓ No. of Blank Lines	NB
✓ Complexity	Cx

Symbols Defined by Henry & Kafura

✓ Length	No. of lines of source code in the procedure
✓ Fan – IN	No. of local flows that terminates at the procedure
✓ Fan – OUT	No. of local flows that emanate from the procedure

Symbols Defined by Thomas McCabe (1976)

✓ Edge	e
✓ Vertices	n
✓ Connected Component	p
✓ Cyclomatic No.	V(G)

Cyclomatic no **V(G)** is equivalent to **e – n + p**. Information Flow is defined by **Henry and Kafura** and it is based upon the No. of lines of source code in the procedure, No. of local flows that terminates at the procedure, and No. of local flows that emanate from the procedure. Information Flow Complexity **IFC** is equivalent to **Length x (Fan-In X Fan-Out)^2.** Comment Percentage will be depended upon No. of Comments, Line of Code and No. of blank lines in code. Comment Percentage **CP** will be equal to **(NC / (LOC – NB)) * 100.** Now Complexity Factor **CX** will be calculated by taking the **Mean of V(G), IFC & CP.**

4.3 Detail Description of Change Effort Factor

Detail description of Change Factor is illustrated in the Figure 6. Change Effort factor will be depended upon Corrective Change and Function Point further Corrective Change Factor will be depended upon Correctness and Reusability. And Function Point will be depended upon Portability and Modification. Table 3 will consist of all the Symbols defined for determining Change Effort. Correctness will be calculated on the bases of requirements fulfilled and the total requirements required in the product. **CN** will be equivalent to **(RF / RT) * 100.** Reusability will be calculated on the basis of No. of Re-Usable Components and the total No. of Components in the product. **RU** will be equivalent to **(CR / CT) * 100.** Now, by this Corrective Change **CC** can be determined by taking **Mean of CN & RU.** Portability will be calculated on the bases of Successful Ports and total No. of Ports in the product. **PB** will be equivalent to **(PS / PT) * 100.** Modification will be calculated on the basis of Time consumed in change process and Total development time of the product. Thus **MF** will be equivalent to **(TC / TD) * 100.**

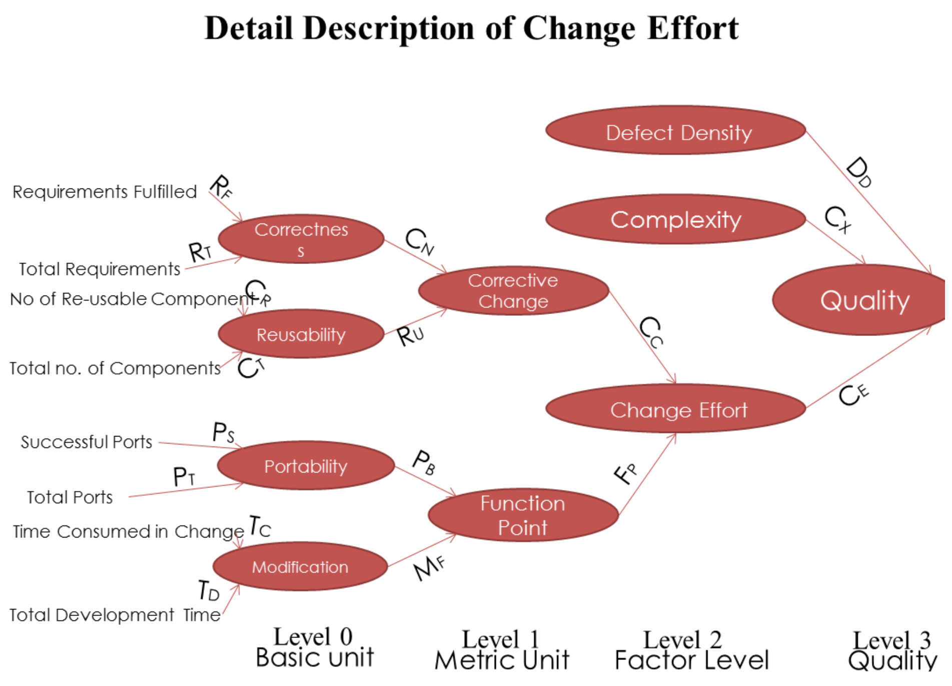
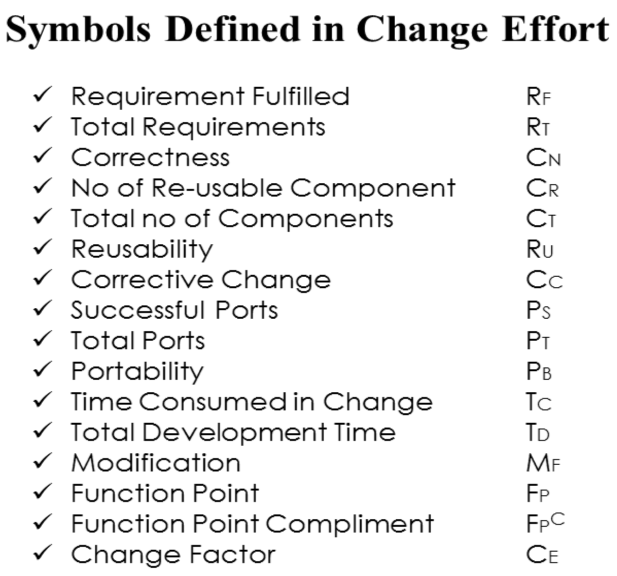


Figure 6: Detail Description of Change Effort

Now, by this compliment of Modification **MF^C** will be **100 – MF.** Function Point **FP** can be determined by taking **Mean of PB & MF^C.** By this Change Effort **CE = Mean of CC & FP.**

Table.3 Symbols Defined in Change Effort



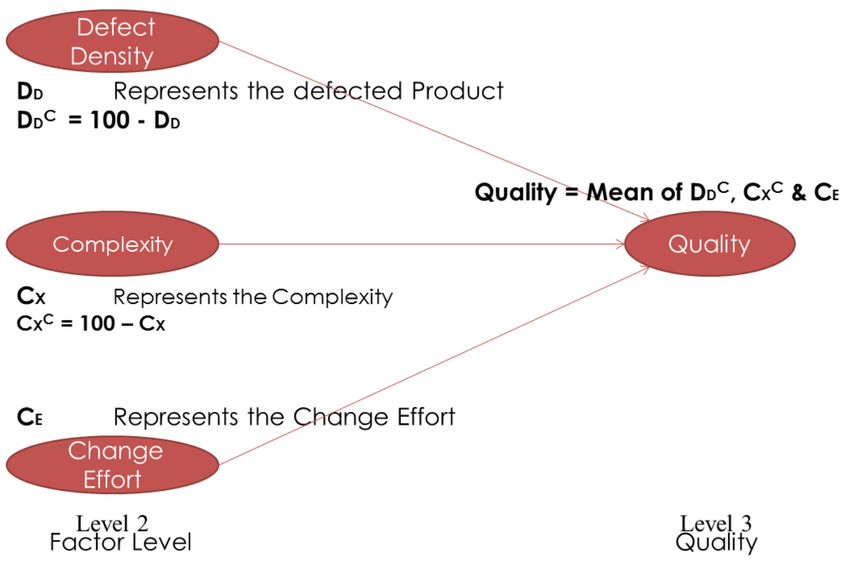
Symbols Defined in Change Effort

✓ Requirement Fulfilled	R_F
✓ Total Requirements	R_T
✓ Correctness	C_N
✓ No of Re-usable Component	C_R
✓ Total no of Components	C_T
✓ Reusability	R_U
✓ Corrective Change	C_C
✓ Successful Ports	P_S
✓ Total Ports	P_T
✓ Portability	P_B
✓ Time Consumed in Change	T_C
✓ Total Development Time	T_D
✓ Modification	M_F
✓ Function Point	F_P
✓ Function Point Compliment	$F_P{}^C$
✓ Change Factor	C_E

4.4 Final Description of Product Quality

At this stage Value of all factors that have been considered in the model are known to us.

Final Product Quality



Defect Density
D_D Represents the defected Product
$D_D{}^C$ = 100 - D_D
Quality = Mean of $D_D{}^C$, $C_X{}^C$ & C_E
Complexity
Quality
C_X Represents the Complexity
$C_X{}^C$ = 100 – C_X
C_E Represents the Change Effort
Change Effort
Level 2
Factor Level
Level 3
Quality

Figure 7: Final Product Quality

All of them can be normalized be eliminating the all decimal places so found. For determining quality we must first find out compliment of Defect Density as product if free from defects. And compliment of Complexity also as Quality of a product is inversely proportional to the complexity factor. Final description of product quality is illustrated in Figure 7. Now, Final Product Quality will be calculated by taking the mean of all three factors.

Quality = Mean of DD^C, CX^C & CE

5. Future Work

In this paper I have proposed and described a new model for finding out quality of a product built by using object oriented approach. This model is a combination of some of the predefined metrics with new approach to find out relationship among them. A detailed comparative study is left out to be done on this model. This will be helpful in determining the strength and weakness of this model. I'm trying to have sufficient experimental results to prove and provide strength to the proposed Quality model.

6. Conclusion

This paper represents a new model for determining the quality of a product built by using object oriented approach. Product quality in this model is based on the factors namely Defect Density, Complexity and Change Effort. No of parameters are used in this for determining various values such as Design Change, Error, Testability, Reliability, Cyclomatic Complexity, Information Flow, Comment Percentage, Correctness, Reusability, Portability and Modification. Defect density depends upon Change Factor and Efficiency. Complexity will depend upon Cyclomatic Complexity, Information Flow and Comment Percentage. Change Effort will depend upon Corrective Change and Function Point.

7. References

[1] Deepak Arora, Pooja Khannaand Alpika Tripathi, Shipra Sharmaand Sanchika Shukla, Software Quality Estimation through Object Oriented Design Metrics, IJCSNS International Journal of Computer Science and Network Security, Vol.11 NO.4, April 2011

[2] J.Bansiya and C.G. Davis, 2002. A Hierarchical Model for Object-Oriented Design Quality Assessment, IEEE Transactions on Software Engineering, Vol. 28, No. 1.

[3] Rizvi S.W.A. And Khan R.A., Maintainability Estimation Model for Object-Oriented Software in Sannella Design Phase (MEMOOD), JOURNAL OF COMPUTING, VOLUME 2, ISSUE 4, APRIL2010.

[4] Kiranjit Kaur, A Maintainability Estimation Model and Metrics for Object-Oriented Design (MOOD). International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, No 5, May 2013

[5] Gentzane Aldekoa, Salvador Trujillo, Goiuria Sagardui and Oscar Diaz, Experience Measuring Maintainability in Software Product Lines, XV Jornadas de ingenieriadel Software y Bases de Datos JISBD 2006.

[6] DagpinarMelis and Jahnke Jens H., Predicting Maintainability with Object-Oriented Metrics-An Empirical Comparison, proceedings of the 10th Working Conferenceon Reverse Engineering (WDRE’03)2003 IEEE

[7] Aditya Jain, A proposed Model for Software Quality with Quality Parameters Specifically for Object Oriented Concept, Chicago International Journal of Advance Studies ISSN 2321-9173, Vol 1, Issue 1, December 2013.