ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 10, October 2016

Analysis of Software Complexity Using Object Oriented Design Metrics in Java Application

Minimol Anil Job

Assistant Professor, Faculty of Computer Studies

Arab Open University, Kingdom of Bahrain

Abstract- Ensuring quality of the software without the knowledge of software evaluation metrics is difficult. Some measurement methodologies are required in completion of software quality evaluation. Software Metrics are essential in software engineering formeasuring software complexity, estimating size and evaluating qualityof the software as well as project efforts. This research paper focuses on evaluation of software complexity metrics applied in object oriented systems by evaluatingsource codes developed in Java. To evaluate the complexity the software metrics used in the research are Lines of Code (LOC) and cyclomatic complexity. The complexity of Java classes in a package is determined using software tools. The results show that these complexity metrics can be used to predict the quality of the software.Software system can be evaluated using software metrics thus quality can be improved.

KEYWORDS- Software quality, Complexity, Cyclomatic complexity, LOC, object oriented metrics

I. INTRODUCTION

In this digital era every business is a digital business: a social, mobile, and web-focused business.The usage of information systems in organizations has increased significantly due to the advancement of IT industry. Software quality estimation has been proved to be one of the most motivating researches in the context of software engineering. To ensure the development of high quality software systems developers need to understand and adopt software quality metrics during development process [1][2]. Product metrics are directly associated with the product thus there is a need to measure quality or characteristics of a product to ensure quality in the developed product. Developing a good software system is a challenging task and software quality assurance is faced with many challenges. Good software metrics must have the ability to measure and predict the quality of the developed software. Software metric is a measurable property which is an indicator of one or more of the quality criteria that we are seeking to measure [1]. This paper assess the quality of software using specific matrices, complexity metrics, applied in object oriented programming by evaluating source codes developed in Java.

II. RELATED LITERATURE

Major customer expectationabout a software product is that the developed software will boost productivity[1][3]. The complex systems we build do work and they demonstrate the skills of the software engineers who build them, and also to the techniques that they use. These techniques are sometimes based on scientific or mathematical rules. The contributions of applicable science and mathematics in the development of software systems are increasing. Some such contributions are finite automata (from discrete mathematics), to describe the behavior of a system; statistics, to show when testing is sufficiently complete; formal methods, for the development of critical components – an example being the use of mathematical notations to specify behaviors unambiguously; metrics, for the measurement of quality attributes. In this section you will study a number of examples of metrics and their use in quality measurement. [4][5]McCall has written very widely on the subject of software quality. McCall and Cavano (1978) identified three general types of requirements that impinge on software product quality; these types still stand today. The three types are: product operation requirements, product revision requirements and product transition requirements.[8][9][10] In addition, McCall described which attributes of a software product, or software quality factors (SQFs) as he termed them, are affected by these three types of requirements, product operation requirements, product revision requirements and product transition requirements[6][7][8].Product operation requirements are Correctness: Reliability: Efficiency, Integrity and Usability. Product revision requirements are Maintainability, Testability and Flexibility. Product transition requirements are Portability, Reusability and Interoperability. Primary Software quality Factors and their measurements are Correctness, Integrity,Maintainability and Usability. [6][9].

This paper assess the quality of software using specific matrices applied in object oriented programming by evaluating source codes developed in Javato measure the complexity of the code.Some of software metrics which are measurable quantities that can be used to infer the values for the SQFs: complexity, consciences,security...etc. Some of them such as complexity can be directly estimated from the code of an implementation. Others, such as training, are subjective and require questionnaires or checklists to be developed and assessed, and calculations made there from.

2.1 System complexity could be measure by two approaches:

- Lines-of-code metric (LOC) is given by counting the number of lines in a piece of code.More lines imply more errors.
- McCabe's cyclomatic-complexity metricmeasures the complexity of method by counting number of independent paths in method. Number of independent paths = number of decision points, each one of the: if, (while, do while, for) loops, switch for each non-default case test, try for each catch block but not the final block, &&, ||, add 1 to cyclomatic complexity, start counting from 1.

2.1 Additional object-oriented complexity metrics

Object-oriented systems require two levels of complexity metric, one to measure method complexity and the other to measure the complexity of the class structure.

Depth-of-inheritance-tree (DIT) metric:DIT is defined as the largest number of hops through an object's superclasses, where the starting class is numbered 0. For a single inheritance programming language like Java, this means that the DIT is the number of ancestors + 1, but in languages like C++ that support multiple inheritances.[12]

Coupling-between-objects (CBO) metric: for a given class, CBO is defined as the number of relationships the class has with other classes. The more relationships a class has, and so the higher the value of this metric, the more difficult it is to understand the use of the given class.[12][13]
Number-of-children (NOC) metric: for a given class, NOC is defined as the number of immediate children for that class. This metric is a measure of the number of classes that will be affected by changes to a given parent class.[12][13]
Response-for-a-class (RFC) metric:for a given class, RFC is defined as the size of the response set for the class, which consists of all the methods of this class (including methods inherited from superclasses), together with all the methods that are invoked on objects of other classes.
Lack-of-cohesion-in-methods (LCOM) metric:for a given class, LCOM measures its cohesiveness. LCOM is defined as the number of pairs of methods that do not make reference to the same attributes, minus the number of pairs of methods that do, or zero should this be negative. In highly cohesive classes, methods will manipulate the same attributes.[13][14]
Weighted-methods-per-class (WMPC) metric: for a given class, WMPC measures its complexity of behavior. It is defined as the sum of the cyclomatic complexities of each method of the class. .[13][14]

The below table shows the Object oriented constructs for the complexity metrics.[12]

Metric	Java Construct
CyclomaticComplexity	Method
Lines of Code	Method
Depth-of-inheritance-tree (DIT)	Inheritance , to find the depth of the tree
Coupling-between-objects (CBO)	Coupling
Number-of-children (NOC)	Inheritance , to find the number of decedents of the class
Response-for-a-class (RFC)	Class/Message
Lack-of-cohesion-in-methods (LCOM)	Class/Cohesion
Weighted-methods-per-class (WMPC)	Class/Method

Table-1: Object oriented constructs and complexity metrics

III. METHODOLOGY

The method adopted in this research is theoretical analysis through related literature and practical analysis using specific software. The researcher studied and evaluated software quality attributes through related literature by reading various books in the discipline and also by referencing related articles published. Object-oriented complexity metrics are analyzed through research from articles and books. The practical approach used in determining the size and complexity using software tools.

Lines of code (LOC): This measure has been found through the Java integrated Development Environment (IDE)- NetBeans. Using a text editor like Notepad or an IDE like NetBeans to write the program, saving the program (source code) in a .java file then compiled using the command javac to create a .class file, which contains the compiled version of the program (Java byte code)

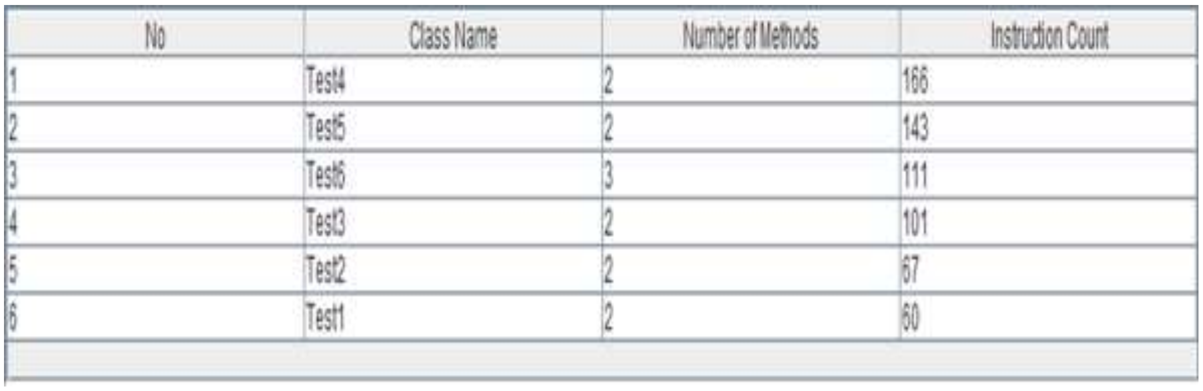
Cyclomatic Complexity Metrics: This measure has been found using the software CyVis in which the researcher found the complexity, total number of methods and number of lines. [15]CyVis is a free software metrics collection, analysis and visualization tool for java based software.CyVis collects data from java class or jar files. Once the data is collected, metrics like number of lines, number of statements, methods and number of decision pointsare determined for the selected classes in a package. Cyclomatic complexity metric can also be determined using the number of independent paths in a method.Once the metrics are collected, the statistical information can be viewed as charts, graphs and tables. The visualization of information is presented in a way, that the developer will be able to know something might be wrong in their developed software.

IV. DATA ANALYSIS AND RESULTS

The complexity metrics used in this research is not directly supporting the measurement of software quality determination. But these metrics help the developers to find the factors affecting the performance of classes in a developed system and thus will be able to suggest improvement mechanisms. Furthermore most complex parts of each class can be determined and suggest more attentive actions to those parts.

Lines of Code (LOC) are the total number of executable lines, comment lines are excluded. Below tables shows the results of experiments for various Java Classes

Package Test includes six classes. Following figure shows the number of methods and number of lines of code in each class.



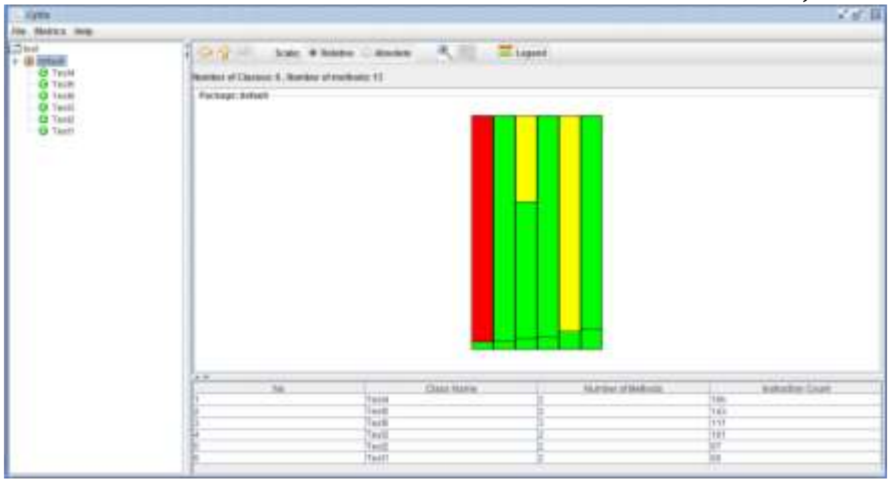
No	Class Name	Number of Methods	Instruction Count
1	Test4	2	166
2	Test5	2	143
3	Test6	3	111
4	Test3	2	101
5	Test2	2	67
6	Test1	2	60

Figure-1 : Lines of Code and Number of Methods in classes in the Package

The package named "Test" used in this practical experiment consisits of 6 classes. The lines of code and number of methods used in each class is computed and presented in Figure-1. Classess Test1, Test2, Test3, Test4, Test5 consists of two ,mehods each and Class Test6 consists of 3 methods. By adding the total number of lines in the claasess total number of lines of code in each package can be determined. Thus total number of lines of code can be determined for the develped software.

<u>The cyclomatic complexity</u> is calculated by counting the number of methods and the complexity in the control flow of lines.

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 10, October 2016

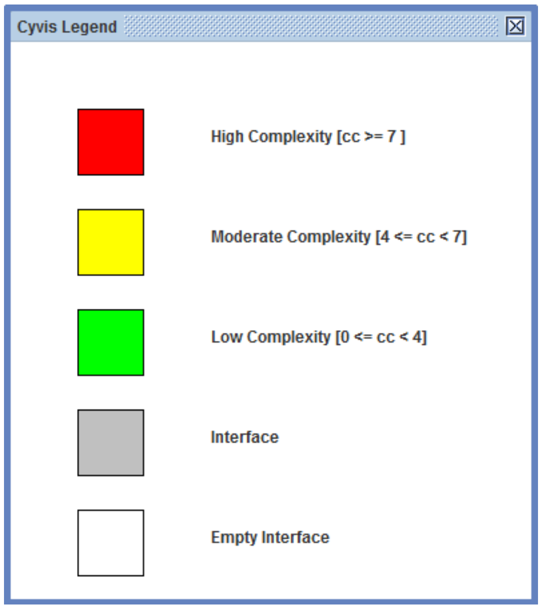


Number of Classes: 6, Number of methods: 11

Package: default

No.	Class Name	Number of Methods	Instruction Count
1	Test4	2	186
2	Test6	2	143
3	Test8	2	117
4	Test3	2	101
5	Test2	2	97
6	Test1	2	89

Figure-2 : Cyclomatic Complexity for the package


Cyvis Legend

🟥	High Complexity [cc >= 7]
🟨	Moderate Complexity [4 <= cc < 7]
🟩	Low Complexity [0 <= cc < 4]
⬜	Interface
□	Empty Interface

Figure-3 : Cyclomatic Complexity Measures Preferences

The six classes' in Package "Test" complexity is evaluated and presented in Figure-2. The vertical bars in the figure repreensts classes and the horizontal lnes represemts methods. The shaded colours in each method shows the cyclomatic complexity of that particular method. Figure-3 shows the Cyclomatic Complexity Measures Preferences. Red colour shows high complexity and it has a value 7 and above. Yellow colour shows moderate complexity and it has a value between 4 and 7. Green colour shows low complexity and it has a value between 0 and 4. The other two colours representas interfecase and empty interface.

Cyclomatic complexity of class Test1 is shown in Figure-4 below. From the figure it is clear that the class has low complexity for the two methods included in the class because both the divisions are in green color.The complexity is low for this class method due to the fact that both methods used in the class has less number of independent paths or number of decision points. That is less of usage of if, (while, do while, for) loops, switch for each non-default case test, try for each catch block but not the final block, &&, || operators in both methods.

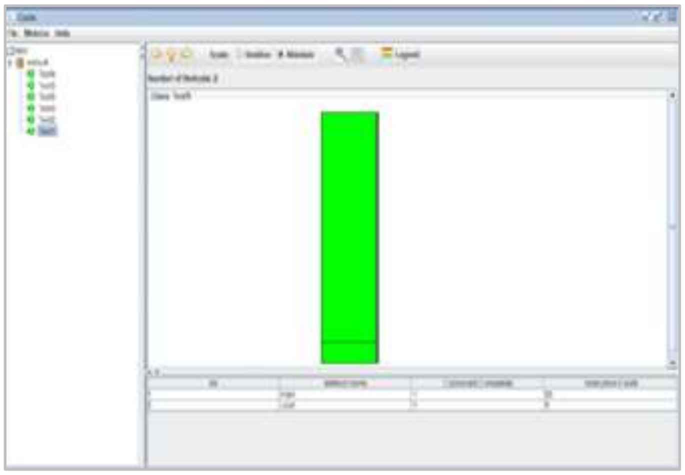


Figure-4 : Class Test-1 Cyclomatic Complexity Visualization

Cyclomatic complexity of class Test2 is shown in Figure-5 below. From the figure it is clear that the class has two methods and the first method has moderate complexity which is shown in yellow color and the other method has low complexity which is shown in green colour.The reason for this difference is the number of independent paths or number of decision points used differently in both methods. That is less of usage of if, (while, do while, for) loops, switch for each non-default case test, try for each catch block but not the final block, &&, ||operators in the lower class method and more usage in the upper class method.

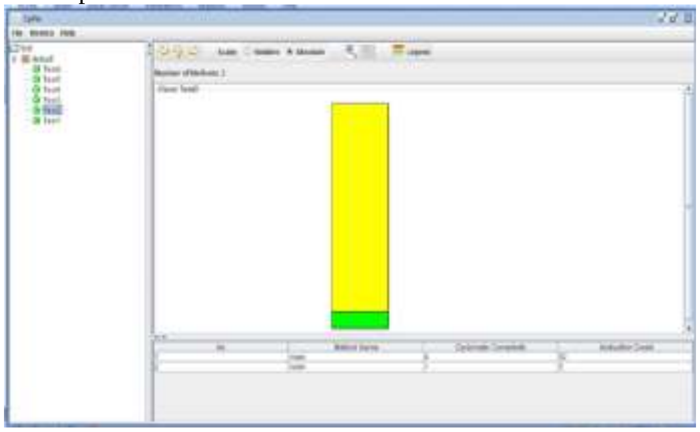


Figure-5 : Class Test-2Cyclomatic Complexity Visualization

Cyclomatic complexity of class Test3 is shown in Figure-6 below. From the figure it is clear that the class has low complexity for the two methods included in the class because both the divisions are in green color.The complexity is low for this class method due to the fact that both methods used in the class has less number of independent paths or number of decision points. That is less of usage of if, (while, do while, for) loops, switch for each non-default case test, try for each catch block but not the final block, &&, || operators in both class methods.

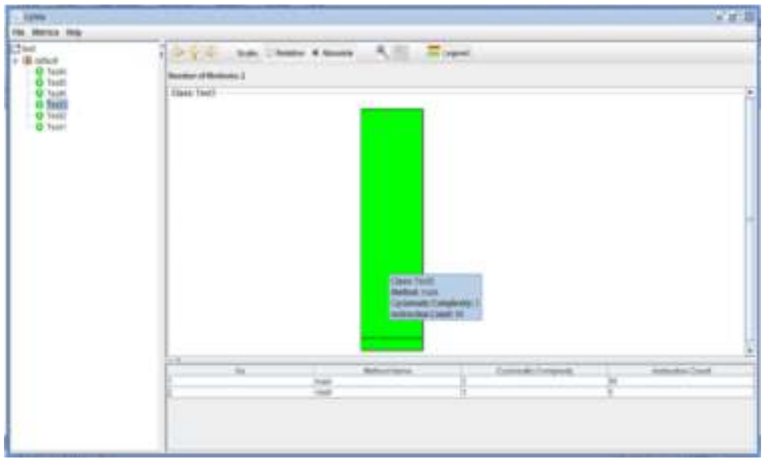


Figure-6 : Class Test-3Cyclomatic Complexity Visualization

Cyclomatic complexity of class Test4 is shown in Figure-7. From the figure it is clear that the class has two methods and the first method has high complexity which is shown in red color and the other method has low complexity which is shown in green colour.The complexity is high in one method because this method uses more number of independent paths or number of decision points. That means more usage of if, (while, do while, for) loops, switch for each non-default case test, try for each catch block but not the final block, &&, || operators in the upper class method and less usage in the lower class method.

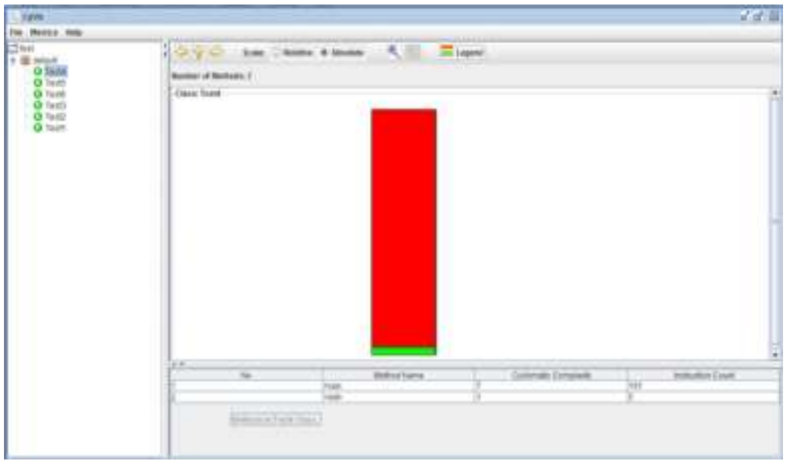


Figure-7 : Class Test-4Cyclomatic Complexity Visualization

Cyclomatic complexity of class Test5 is shown in Figure-8. From the figure it is clear that the class has low complexity for the two methods included in the class because both the divisions are in green color.The complexity is low for this class method due to the fact that both methods used in the class has less number of independent paths or number of decision points. That is less of usage of if, (while, do while, for) loops, switch for each non-default case test, try for each catch block but not the final block, &&, || operators in both methods.

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 10, October 2016

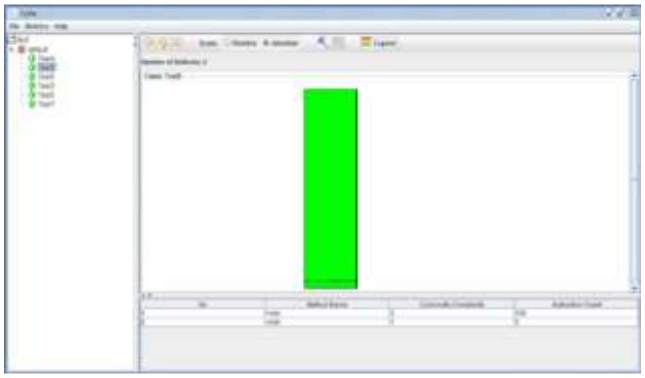


Figure-8 : Class Test-5Cyclomatic Complexity Visualization

Cyclomatic complexity of class Test6 is shown in Figure-9. From the figure it is clear that the class has three methods and the first method has medium complexity which is shown in yellow color and the other two methods have low complexity which are shown in green colour.The reason for this difference is the number of independent paths or number of decision points used differently in both methods. That is less of usage of if, (while, do while, for) loops, switch for each non-default case test, try for each catch block but not the final block, &&, || operators in the lower class methods and more usage in the upper method.

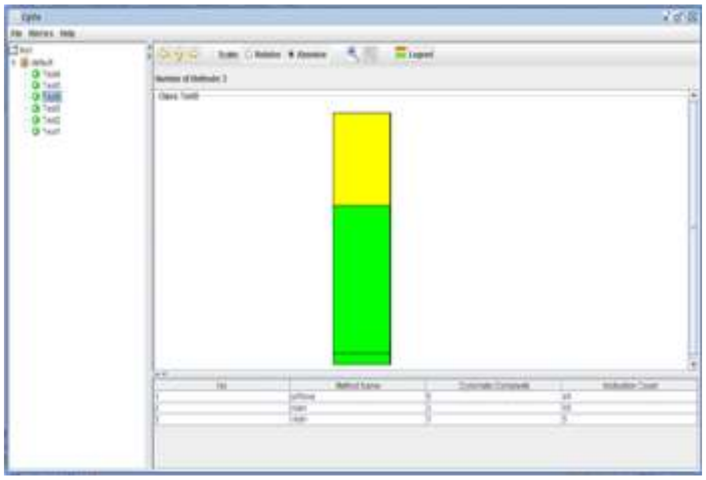


Figure-9 : Class Test-6Cyclomatic Complexity Visualization

V. CONCLUSION

In this paper software quality metrics and the methods to measure them are observed as well as evaluated using software tools. Software measurement and metrics help us a lot of evaluating software process as well as the software product. Since the research is based on the object oriented software metric evaluation, Java classes are used for observation. The results show that these complexity metrics can be used to predict the quality of the software. Lines of code will help in software maintenance. The cyclomatic complexity metric shows which part of the program code has more complexity and this will help the developers to give more attention to those parts. Well-designed metrics with documented objectives can help developers to improve its software product, processes, and customer services.A future research is recommended to evaluate the other Object-oriented Complexity metrics which will help the software developers to ensure the quality of the software.

REFERENCES

[1] Alan Gillies, Software Quality: Theory and Management (3rd edition) Paperback – January 17, 2011

[2] José P. Miguel , David Mauricio, Glen Rodríguez, A Review of Software Quality Models for the Evaluation of Software Products ,International Journal of Software Engineering & Applications (IJSEA), Vol.5, No.6, November 2014

[3] Thapar SS & Singh P & Rani S. (2012) "Challenges to the Development of Standard Software Quality Model," *International Journal of Computer Applications* (0975 – 8887) Volume 49– No.10, pp 1-7.

[4] Klas Michael &Constanza Lampasona &Jurgen Munch. (2011). "Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results," *37th EUROMICRO Conference on Software Engineering and Advanced Applications*, 978-0-7695-4488-5/11, IEEE pp. 341-348

[5] PensionwarRutuja K & Mishra Anilkumar& Singh Latika. (2013). "A Systematic Study Of Software Quality – The Objective Of Many Organizations, "*International Journal of Engineering Research & Technology (IJERT)*, Vol. 2 Issue 5

[6] Al-BadareenAnas Bassam. (2011). "Software Quality Evaluation: User's View," *International Journal of Applied Mathematics and Informatics*, Issue 3, Volume 5, pp 200-207

[7] Ghayathri J &Priya E. M. (2013) "Software Quality Models: A Comparative Study," *International Journal of Advanced Research in Computer Science and Electronics Engineering* (IJARCSEE) ,Volume 2, Issue 1, pp 42-51

[8] Galin, Daniel (2004), Software Quality Assurance – From theory to implementation, Pearson –Addison Wesley, England

[9] HuaiLiu ,Fei-ChingKuo , TsongYueh Chen "Teaching an End-User Testing Methodology"Software Engineering Education and Training (CSEE&T), 2010 23rd IEEE Conference on 9-12 March 2010

[10] A Literal Review of Software Quality Assurance,C. SenthilMurugan S. PrakasamPh.D.,International Journal of Computer Applications (0975 – 8887) Volume 78 – No.8, September 2013

[11] Amit Sharma1, Sanjay Kumar Dubey2, IJCSMS International Journal of Computer Science & Management Studies, Special Issue of Vol. 12, June 2012ISSN (Online): 2231 –5268, www.ijcsms.com, Comparison of Software Quality Metrics forObject-Oriented System

[12] C. Neelamegam, M. Punithavali, "A survey onobject oriented quality metrics", Global journal ofcomputer science and technologies, pp 183-186,2011.

[13] A. Deepak, K. Pooja, T. Alpika, S. Sharma,"Software quality estimation through objectoriented design metric JCSNS Internationaljournal of computer science and network security , April 2011, pp 100-104.

[14] Gurdev Singh. et al.," A Study of Software metrics", IJCEM International Journal of Computational Engineering & Management, Vol. 11, January 2011

[15] cyvis.sourceforge.net

Author's Biography

Dr Minimol Anil Job is working as an Assistant Professor in ITC Department at Arab Open University's Bahrain Branch. She has more than twelve years of academic experience in the field of Information Technology. Her research interests include Software Engineering, Database management, big data analysis and cloud computing