Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 5, May 2015, pg.99 – 106

RESEARCH ARTICLE

Improving the Design of Cohesion and Coupling Metrics for Aspect Oriented Software Development

Mandeep Kaur[1], Department of Computer Science and Technology, Lovely Professional University, Punjab, India
comp.mandeep@gmail.com

Rupinder Kaur[2], Department of Computer Science and Technology, Lovely Professional University, Punjab, India
rupinder.16835@lpu.co.in

Abstract— Software metrics play an important role in determining the quality of software. There are various attributes of quality that need to be understood for developing better quality software. These factors include coupling, cohesion, complexity, maintainability, testability etc. Hence metrics are required to calculate the values for all these quality attributes. This study focuses on developing metrics for better calculation of coupling and cohesion values.
Over the recent years, Object Oriented Programming System (OOPS) has become popular and completely replaced the Procedural Oriented Programming. OOPs implementing complex software systems and become great success in modeling. But OOPs also has some limitations like in the system decomposition there are some functionalities cannot be assigned to single module. There is one paradigm that enhances software design and promotes reusability called Aspect Oriented Paradigm (AOP).So we focus on several object oriented metrics and find how Aspect affect these metrics.

I. INTRODUCTION

Software metrics have become very important in the discipline of software engineering. Software quality and cost of the project can be measured by using these metrics. In the
stream of software growth, metrics can be used for secured fragments of software systems as well as recognizing wherever refactoring can applied. And observe that the quality is expands or reduces in the structure of the evolution of software systems. In the area of software reverse engineering the estimation of quality and complexity of the software system can be measured by metrics, to get a basic knowledge and deliver hints about delicate portions of software systems.

Outmoded Measures of Complexity

In late 90's, the software measurement techniques was proposed and the proposed technique is the Source Lines of Code (SLOC) metric, which is quiet used now a days. In the software program it is used to measure the amount of code. Especially the amount of effort and estimate the cost of development of the program and also estimate the productivity of the produced software .There are two types of SLOC one is Physical and another is logical. The physical SLOC is used to calculate the none comment lines in the copy of the program's source code. Logical SLOC is used to calculate the number of statements.
There are amount of deficiencies of expending an unprocessed measures for example LOC are used as a alternate measure for different ideas of program size as functionality and complexity. As the diversity of programming language is increases day by day, so there is a need of more discriminating measures, as LOC in an assembly language and it cannot be equivalent in functionality or complexity to LOC in a high-level language [7].

Coupling

Stevens etal. Structured development techniques were first introduced in the context of coupling. They measurement of the strength of association is defined by coupling that is recognized by a connection between one module to another. They stated

that if the modules are related to each other than the coupling is more stronger between the modules and it is difficult to make changes in more complex software system.
Myers [10] improves the coupling by explaining different coupling levels. However coupling could only be firm by hand as the definitions were neither exact nor narrow, leaving room for subjective interpretations of the levels.

Cohesion
In cohesion the single components are required to attain the same task. It was introduced within the environment of same design. The module is stated by reviewing the relationship between the pairs of its processing elements. This term was defined as an action performed by a module. In other informal definition the ordinal scale is developed for measurement that describes the amount to which any action performed by the module contributed to the unified function [11]. Basically there are seven kinds of it. The most desirable to the least desirable.

II. STATIC AND RUNTIME METRICS

There are so many metrics proposed to calculate the quality of the object oriented design. There are two categories in which design metrics are classified: 1) static 2) dynamic/runtime. The task of static metrics is to measure, what happen when execution of program takes place and also measure the quantity and complexity of different features of the source code. The task of run time metrics is to measure actually what happen when there is an execution of program take place. They find the run time behavior, complexity and also find the characteristics of source code. Without affecting the structure of research work, the developers are able for developing or designing the quality metrics. In compare of static metrics the analysis of run time metrics is very expensive and very complex in case of performance [12]. For the object oriented design environment the developer giving less importance to the run time metrics. There are so many situations which the static coupling and cohesion does not measure the real situation of the software these cases like dynamic binding, polymorphism and unused code present in software while it is less related to the classes at run time. Most of the real time applications have very complex and dynamic behavior that provide a strong reason to move our attention towards run time metrics instead of existing static metrics. Now our work is to find out whether run time measures of coupling and cohesion providing the more useful information in the object oriented design in compare to the information given by the simple static measures. Then only we will decide that it is beneficial to continue our research in the field of run time cohesion and coupling metrics or further more to find out their relationship with the external quantity [13].

III. OBJECT ORIENTED METRICS

In the development of software today's, the object-oriented development is becoming very common. There is a different methodology to design the object-oriented development and also requires a changed method to software metrics. Since the objects are used in object oriented technology and the fundamental building blocks are algorithms, object oriented program are using different approach for software metrics from the standard metrics set. For traditional functional/ procedural programs, lines of code and cyclomatic complexity are used as the metrics and become accepted as standard and were used to form an idea of object oriented environments at the opening of the object-oriented design uprising. However, procedural languages using outmoded approach are not suitable for assessing object oriented software, mostly because the basic elements like polymorphism inheritance, classes and object are not measure to design. The analyze of the object-oriented software are only used to internment a small part of such software and thus provide a fragile quality suggestion[8] [9]. In the literature many object oriented metrics are proposed. Now the question arises is that which type of proposed metrics is used in our project. With the comparison of other software, the object oriented software is more complex concept in which there is not a single software quality is measured. The developer must define the characteristics of software to improve the software quality and then decide how the quality of the software is to be measure. For others who can understand your view point you can make the quality measured in a very easy way. Coupling and cohesion is used to calculate the seminal methods of object oriented design.

IV. ASPECT ORIENTED PROGRAMMING

Aspect-Oriented Programming (AOP) is the improved version of object oriented programming (OOP). AOP is the technique which giving the programmer a different way of structuring the program. OOPs were basically structured around class. As the same way AOP is structured around aspects. Aspect gives authority to which a system component may be separated or combined such as transaction management.
In software engineering, Aspect oriented programming is the different method in which functional, procedural, are introduced by aspect oriented programming in different way. This technique does not replace these concepts but improving the ability if these concepts. The basic idea of aspect oriented programming is to provide clean separation and combination of component by the use of aspects.

Development of Aspect Oriented

As the name suggests Aspects are particular part or features, as we are working with the aspect oriented it must be fall under initial development. They use the wide range of programming language for the designing of the system, static analysis of the system and one more thing for requirement specification. In this we neglect the migration part.

V. LITERATURE REVIEW

Divya Sharma, PoojaNarula [1] proposed that a good quality software product requires efficient measures to accurately monitor the internal software quality attributes. Software metrics have been widely and successfully used to measure such internal quality attributes for object-oriented software systems. Coupling is an important qualitative measure for measuring the performance of software. Coupling is defined as how much dependent a module is on other modules. Coupling defines the external complexity of a class, i.e. how dependent a class is on other classes.

For coupling measurement there are two metrics available static metrics and dynamic metrics. Static metrics measure the expected coupling behavior of object-oriented software and not the actual behavior. Whereas dynamic metrics can measure the actual coupling behavior as they are evaluated from data collected during runtime. Dynamic metrics can be defined as the metrics used to measure the internal quality attributes of object-oriented systems by working on the information gathered from the runtime behavioral analysis of such systems. The objective of my thesis is to propose a new dynamic coupling metrics.

The conventional static metrics have been found to be inadequate for modern object oriented software due to insufficient contribution of object-oriented features. Due to

this fact we need to focus on dynamic metrics in addition to the traditional static metrics. A few new dynamic metrics are proposed for the measurement coupling at run-time. Moreover, different approaches for the dynamic analysis of programs required for collection of run-time data for the measurement of dynamic metrics are compared. AOP approach is used for the computation of coupling metrics. In this paper a dynamic coupling tracer application is developed for the purpose of computation of new dynamic coupling metrics. Existing dynamic coupling metrics take into account only method-method invocations between objects of classes at runtime.

Sandip Maland Kumar Rajnish [2] proposed that Cohesion is an Object-Oriented (OO) software design property that helps for the measuring of degree of interdependency or connectivity within subsystems of a system. Numerous class cohesion metrics can be found in the literature. Which metric is best suited for a given situation is always a critical question. Few metrics are validated empirically against open source software projects. The purpose of this paper is to validate empirically of the proposed new class cohesion metric (CC) using some open source software projects and find the effected quality factors. Results of this paper conclude that CC continuously gives better correlation with Number Line of Code (NLOC) compare to other existing cohesion metrics. The average value of CC (CohS) of a system also predicts the natures (understandability, modifiability, and maintainability) of a system.

An attempt has been made to propose a new cohesion metric which is based on formal definitions, properties of classes. In addition to the proposal, this paper has also presented empirical data on CC and CohS from five open source software projects. All systems have developed in java. From Table 7 and Table 8, it is found that there is a strong correlation between CC and reusability in cases, Pearson correlation and linear correlation. So, this study clearly provided that CC is the valid indicator of external quality attributes of the classes of projects such as reusability. The mean value of CC of a system (CohS) indicates that s*ystem3* and s*ystem4* are more understandable, modifiable, and maintainable than the *system1, system2* and *system5*. This firmly believes us that this work will encourage other researchers and developers to use the results obtained from this study to predict and measure several other software quality attributes.

Girish K. K [3] explained that High cohesion is a desirable property of software as it positively impacts understanding, reuse, and maintenance. Currently proposed measures for cohesion in Object-Oriented (OO) software reflect particular interpretations of cohesion and capture different aspects of it. Existing approaches are largely based on using the structural information from the source code, such as attribute references, in methods to measure cohesion. This paper proposes a new measure for the cohesion of classes in OO software systems based on the analysis of the unstructured information embedded in the source code, such as comments and identifiers. The measure, named the Conceptual Cohesion of Classes (C3), is inspired by the mechanisms used to measure textual coherence in cognitive psychology and computational linguistics. This paper presents the principles and the technology that stand behind the C3 measure. An open source eclipse plug-in is developed as part of this research work to demonstrate how well cohesion of classes is predicted using C3 metrics.

This paper defines the conceptual cohesion of classes, which captures new and complementary dimensions of cohesion compared to a host of existing structural metrics. My major findings are,

1) Conceptual cohesion is a highly advantageous, most relevant aspect of software cohesion which was concealed due to overwhelming usage of structural cohesion.

2) Current industrial settings need a powerful software cohesion metrics that can save software developers as well as software testers bug identification time and debugging time.
3) Most of the structural metrics used today will not incorporate an important aspect of object oriented concept- objects interrelate real world scenarios, which can be solved using Conceptual Cohesion of Classes(C3) metrics.
4) Conceptual Class Cohesion emphasis on the object (real world scenario) and the object itself is the 'concept' in Conceptual cohesion.
5) Comments and identifier names, even though appears static in a source code file, are the best source of information supply. In other words we can say that they are the whole sale suppliers of information.
6) Latent Semantic Indexing (LSI) without any substitute is the most efficient information retrieval approaches available today

SushmaYadav, Dr. Sunil Sikka, UttpalShrivastava [4] explained Software metrics are essential to improve the quality of software during the development process. Coupling and cohesion measures are used in various activities such as impact analysis, assessing the fault proneness of classes, fault prediction, re-modularization, identifying of software component, design patterns, assessing software quality etc. Low coupling and high cohesion are better for good software quality. Coupling and cohesion metrics can be applied at the early phase of the software development process. This paper reviews various coupling and cohesion metrics for object-oriented software. Various coupling and cohesion metrics have been proposed by many researchers in the literature. Coupling and cohesion metric can be classified into two categories – static and dynamic. The static software metrics are obtained from static analysis, whereas the dynamic software metrics are computed on the basis of data collected during run time execution of the software. Static metrics are easy to calculate while dynamic metrics are tough to calculate. Static metrics are useful when we need result from small programs. Ability of static metric is to quantify various aspects of the design complexity or source code of a software system, make them useful in software engineering. Static metrics are inefficient to deal with object-oriented features such as polymorphism.
Heba A. Kurdi [5] explained that AOP showed the capacity to developing something new for the some properties which are nonfunctional components such as exception handling, fault tolerance, logging. These properties represent some of the critical issues that are not efficiently handled by the other programming approach and these approaches are leading to the complex code. The concept of AOP is discussed in this paper. Also discussed about the quality and efficiency of the development of the software and codes.AOP is giving good signs of success in future so the researchers giving more attention towards this technique.
The basic aim of AOP is to offer good system component that may be separated or combined of the code. This paper also gives brief description about the developing technique in AOP that is starting with some use of examples and definitions. All these things need the demand of improving the quality of the software. After that, this paper also discussed the benefits and limitation of AOP and all these are based on the code size, cognition, efficiency, performance, and language mechanism.
The results of this paper do not prove the above limitations except in two measures. These measures are code size and language mechanism. This paper also discussed possible challenges and thread. AOP technique is the difficult process in term of debugging and need more knowledge towards core modules. The conventional programming does not focus on these issues like crosscutting.
A common conclusion was researched, declaring the impact and the need of further deep studies also much more research of AOP, finally indicates that the approach is still new and unpopular. However, the developers who used this approach are confident and they talk assertively about its enhancement to software quality. The empirical studies, though, had another thing to say, and it was not always in favor of AOP.
To final conclusion, it has been noticed that AOP is a very interesting concern that needs to take its righteous place in the programming community. Only then could researchers study AOP effectively and efficiently.

VI. PROPOSED SYSTEM

Cohesion metrics and class level and object level coupling are very helpful in the improvement of quality of object oriented application. The idea of this type of measurement
is produced from the existing measures. Coupling and Cohesion are considered to be the most important attributes. Coupling and cohesion are the attributes which measure the degree or the strength of interaction and relationships among elements of the source code, for example classes, methods, and attributes in object-oriented (OO) software systems. One of the main objectives behind Object Oriented analysis and design is to implement a software system where classes have high cohesion and low coupling amongst them. The Objectives are based on very significant factors of complexity measurement of software which in Coupling and cohesion.

- To study the existing coupling and cohesion metrics.
- To study the importance of Aspect Orientation on object orientated metrics.
- To enhance the object oriented metrics for Aspect Oriented programming.

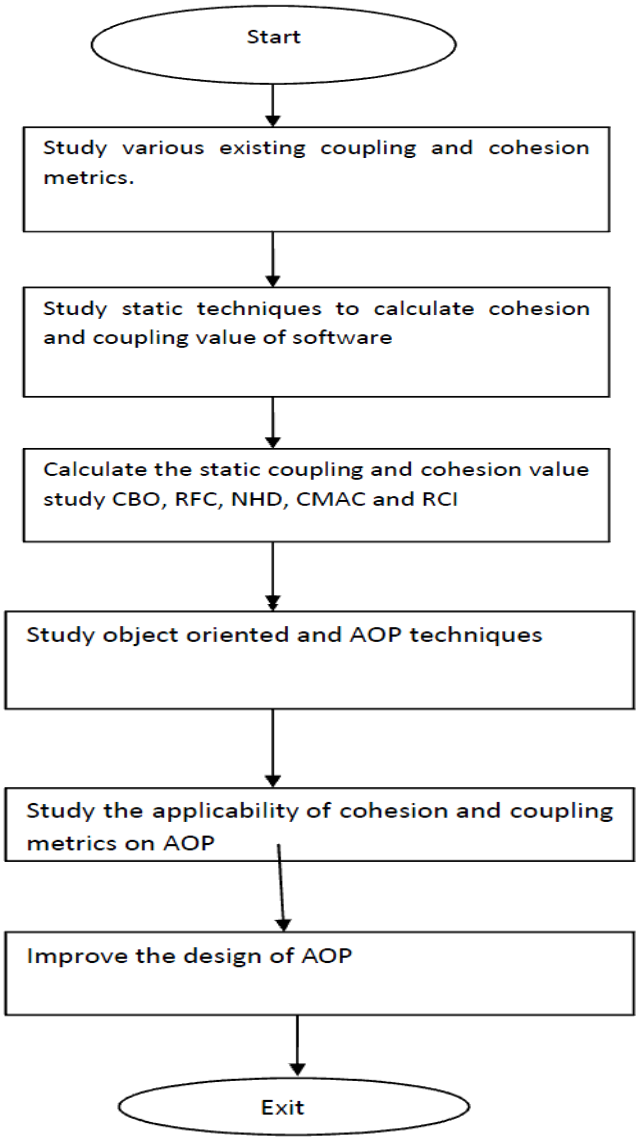

Fig 1: Flow diagram of proposed System

In this work, we are applying static metric on the distributed software systems to calculate the coupling values dynamically. To do so, CBO (Coupling between objects), RFC (Response for Class), CAMC (Cohesion among method of class), NHD (Normalized hamming distance)metrics will be applied in existing metrics using coupling and cohesion metric approach to define the accurate estimated at design level. In present research using static metrics and also on applied static approach to improve the accuracy at design level and to take into better consideration. On the basis of these metrics we will able to maintain relation between class, functions and methods of the software. To drive relationship between all these we will propose metrics. These metrics we are able to calculate cohesion and coupling value of the software. The following method will be used to conduct the proposed work:

- Study various static metric like CBO (Coupling between objects), RFC (Response for class), NHD (Normalized hamming distance), CMAC (Cohesion among method of class) and RCI (Ratio of cohesion interaction) etc.
- Study the significance of AOP (Aspect oriented Programming) on object oriented metrics as AOP is the extended version of the object oriented technique.
- Study the applicability of coupling and cohesion metrics available for object oriented programming system, on AOP.
- Improving the design of AOP (Aspect Oriented Programming).

VII. RESULTS

In this section we will discuss the implementation of proposed technique. There are five types of metrics on which we are working. These metrics are Maintainability Index, Cyclomatic Complexity, Depth of Inheritance, class coupling, Lines of code. We integrate this metric with the aspects and aspects does not replace these concepts but improving the ability if these concepts.

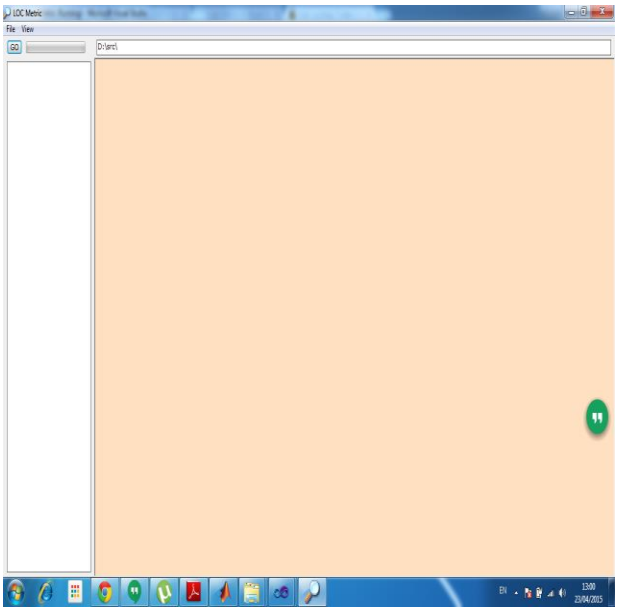

Fig 2: User interface for the calculation of Lines of code

This is the user interface; this interface is designed for the purpose of calculating the lines of codes in the project. In this interface, there are one option given where we can given the path of any project and then we apply go button and then the system calculate the number of lines of code present in the project.

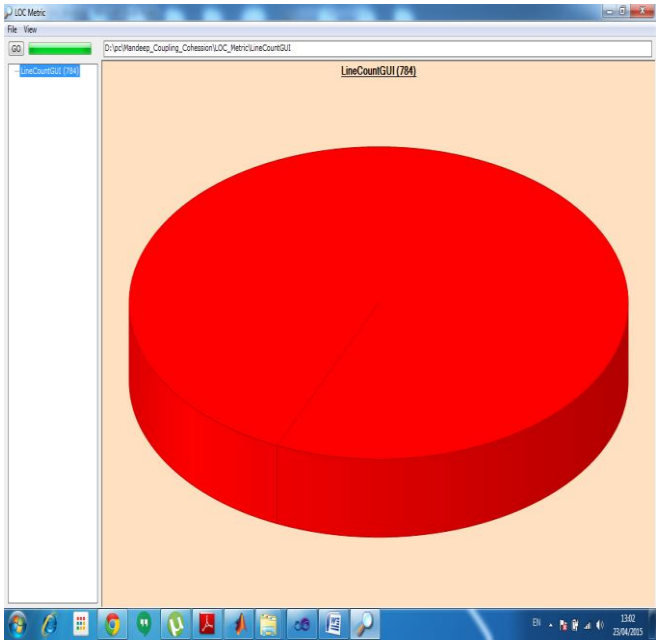

Fig 3: calculation of Lines of codes.

.

This is the dynamic interface in which we can given any project path and then the system calculate the lines of codes present in the project.

There is software metric which is used for the measurement of the computer program in size. This can be done by counting the lines present in the source code of the program.

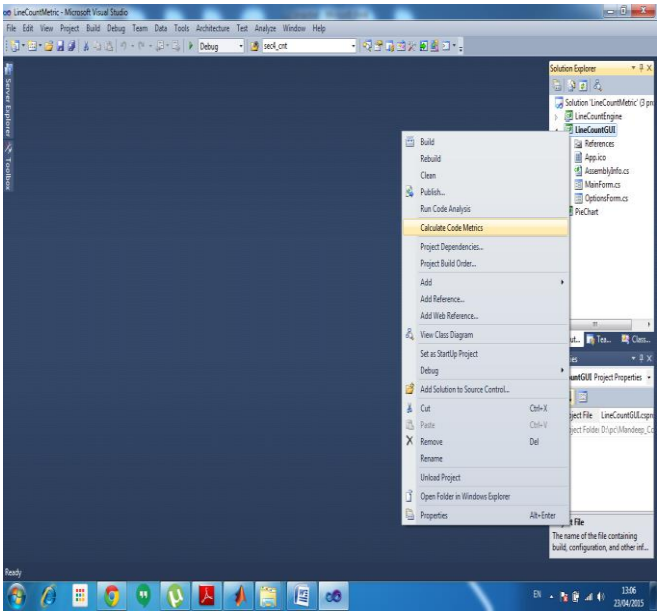

Fig 4: calculation of metrics using .net tool

The .Net tool provide the facility to calculate the code metrics of the given program which include the calculation of the Maintainability Index, Cyclomatic Complexity, Depth of Inheritance, class coupling, Lines of code.

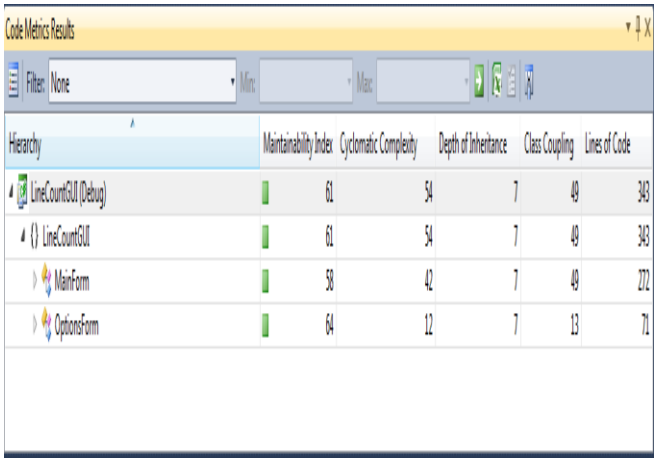

Fig 5: Results of different metrics

Here the results of different metric are generated. Here the system generates the two types of results. The first is system generates the metrics results of complete project that is
Maintainability index = 61
Cyclomatic complexity = 54
Depth of Inherence (DOI) = 7
Class Coupling = 49
Lines of Code (LOC) = 343
Second type of result is generated by the system is generating the metrics results of each file individually.
Here for the file MainForm the system generates the metrics results
Maintainability index = 58

Cyclomatic complexity = 42
Depth of Inherence (DOI) = 7
Class Coupling = 49
Lines of Code (LOC) = 272
For the file OptionsForm the system generates the metrics results
Maintainability index = 64
Cyclomatic complexity = 12
Depth of Inherence (DOI) = 7
Class Coupling = 13
Lines of Code (LOC) = 71

VIII. CONCLUSION

Separation into concerns are past to the process of coming into existence in case of aspect oriented paradigm. While the software is developing the aspect oriented programming giving new mechanisms that will support improvement in the crosscutting concern and become the visible paradigm in the development of aspect oriented software. The impact of this type of programming will be on the metrics of object oriented software is presented. The aspect oriented software give better much better result while comparing with object oriented programming. But this type of programming is very complex while designing the system.

References

[1] Divya Sharma, PoojaNarula , "*Static and Dynamic Analysis of Object Oriented Systems*" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 6, June 2014 ISSN: 2277 128X
[2] Sandip Mal and Kumar Rajnish , " *New Class Cohesion Metric : An Empirical View* " International Journal of Multimedia and Ubiquitous Engineering Vol.9, No.6 (2014), pp.367-376
[3] Girish K. K, "Conceptual Cohesion of Classes(C3) Metrics", *International Journal of Science and Research* (IJSR) ISSN (Online): 2319-7064
[4] SushmaYadav, Dr. Sunil Sikka, UttpalShrivastava, "A Review of Object-Oriented Coupling and Cohesion Metrics" , *International Journal of Computer Science Trends and Technology* (IJCST) – Volume 2 Issue 5, Sep-Oct 2014.
[5] Heba A . Kurdi , " Review on Aspect Oriented Programming" , *International Journal of Advanced Computer Science and Applications*, Vol. 4, No. 9, 2013.
[6] R.R. Gonzalez. "A united metric of software complexity: Measuring productivity, quality and value". *The Journal of Systems and Software*, 29(1):17{37, 1995.
[7] N .E .Fenton and M . Neil . "Software metrics: Successes, failures and new directions". *The Journal of Systems and Software*, 47:149{157, 1999.
[8] T .McCabe. "A software complexity measure". *IEEE Transactions on Software Engineering*2(4):308{320, 1976.
[9] A.J. Albrecht. "Measuring application development". In *IBM Applications Development joint SHARE/GUIDE* symposium, pages 83{92, Monterey California, USA, 1979.
[10] G . Myers. "Reliable Software Through Composite Design". *Mason and Lipscomb Publishers*, New York, USA, 1974.
[11] L.L Constantine and E. Yourdon. "*Structured Design" .Prentice-Hall, Englewood*Cli s, New Jersey USA, 1979.
[12] M .Page - Jones."The Practical Guide to Structured Systems Design". *Yourdon Press*, New York, NY, 1980.
[13] D .A .Troy and S .H. Zweben." Measuring the quality of structured designs".*The Journal of Systems and Software*, 2:112{120, 1981.

