Design of COTS Metrics based upon CK Metrics

Sonu Dhariwal, Ashwani Kumar, Pradeep Kumar Bhatia
Guru Jambeshwar University of Science & Technology Hisar

Abstract

Component Based Software Engineering (CBSE) aims to build software from pre-exiting components, build components as reusable entities and evolve application by replacing components. In this paper, we introdues new modally technique for analyzing the quality of component design and metrics adopted for component based systems. The primary goal of this work on the properties coupling, interface, complexity in figure an possible attributes and weakness and development metrics methologies to measure the quality productivity.

1. Introduction

Component-Based Software Engineering (CBSE) is a methodology that emphasizes the design and construction of computer–based systems using reusable software components. This principle embodies an element of "buy, don't build" that shifts the emphasis from programming software to composing software systems (Pressman, 2001). It is also an approach for developing software that relies on software reuse and it emerged from the failure of object-oriented development to support effective reuse. The behavior and the stability of an application cannot be assessed unless it is tested comprehensively. The quality of the application is high when it yields the expected results, is stable and adaptable and leads to reduce maintenance costs. If a change has been introduced in a component, which has been integrated in an application, the impact of the change on the whole application has to be determined by the developer to assess the stability of the application. Consequently, there is certainly a need to measure quality and assess the component's impact on the overall system. Metrics are needed to measure several types of quality issues [1].

A software component is a coherent package of software implementation that offers well-defined and published interfaces, is reusable and that can be independently developed and delivered; such components are put together to form an application [2]. However, there are no good metrics available to validate their effectiveness, when components are integrated together to form a complete system. Due to the inherent differences in the development of component based and non component based systems, the traditional software metrics prove to be inappropriate for component-based systems.

component based systems, the traditional software metrics prove to be inappropriate for component-based systems.
The component metrics alone are not sufficient for an integrated environment, because there is a need to measure the stability and adaptability of each component when it is integrated with other components.

2. Object- Oriented Metrics

The software metrics for object oriented programs have been provided by Chidamber and Kermerer. They have defined the following six metrics for object oriented design [3].

2.1.1 Weight Methods per Class (WMC)
2.2.2 Depth of Inheritance Tree of a class (DIT)
2.2.3 Coupling between Objects (CBO)
2.2.4 Number of Children (NOC)
2.2.5 Response for a Class (RFC)
2.2.6 Lack of Cohesion in Methods (LCOM)

2.1.1 Weight Method per Class (WMC)

- This metric is used to measure the understability, reusability and maintainability [4].
- A class is a template from which objects can be created. Classes with large number of methods are likely to more application specific, limiting the possibility of reuse [5].
- This set of objects shares a common structure and a common behavior manifested by the set of methods.
- The WMC is a count of the methods implemented within a class or the sum of the complexities of the methods. But the second measurement is more difficult to implement because not all methods are accessible within the class hierarchy because of inheritance.
- The larger the number of methods in a class is the greater the impact may be on children, since children inherit all of the methods defined in a class [6].

2.1.2 Depth of Inheritance Tree (DIT)

- Depth of class within the inheritance hierarchy is the maximum length from the class node to the

root of the tree, measured by the number of ancestor classes.

- The deeper a class within the hierarchy, the greater the number of methods and is likely to inherit, making it more complex to predict its behavior [6].
- A support metric for DIT is the number of methods inherited.

2.1.3 Number of Children (NOC)

- The number of children is the number of immediate subclasses subordinates to class in the hierarchy.
- The greater the number of children, the greater the parent abstraction [5].
- The greater the number of children, greater the reusability, since the inheritance is a form of reuse [7].
- If the number of children in class is larger than it require more testing time for testing the methods of that class [5].

2.1.4 Coupling between Object Classes (CBO)

- Coupling is a measure of strength of association established by a connection from one entity to another [8].
- Classes are couple in three ways. One is, when a message is passed between objects, the object are said to be coupled. Second one is, the classes are coupled when methods declared in one class use methods or attributes of the other classes. Third on is, inheritance introduced significant tight coupling between super class and subclass.
- CBO is a count of the number of other classes to which a class is coupled [9]. It is measured the counting the distinct non inheritance related class hierarchy on which a class depends.
- Excessive coupling is detrimental to modular design and prevent reuse. If the number of couple is larger in software than the sensitivity to changes in other in other parts of design.

2.1.5 Response for Class (RFC)

- A message is a request that an object makes to another object to perform an operation. The operation executed as a result of receiving a message is called a method.
- The RFC is the total number of all methods within a set that can be invoked in response to message sent to an object. This includes all methods accessible within the class hierarchy.
- This metrics is used to check the class complexity. If the number of method is larger that can be invoked from class through message than the complexity of the class is increase.

2.1.6 Lack of Cohesion of Methods (LCOM)

- LCOM uses variable or attributes to measure the degree of similarity between methods.
- We can measure the cohesion for each data field in a class; calculate the percentage of methods that use the data field.
- Average the percentage, then subtract from 100 percent. Lower percentage indicates greater data and method cohesion within the class.
- High cohesion indicates good class subdivision while a lack of cohesion increases the complexity [8].

3. COMPONENT METRICS SUITE FOR CK METRICS

3.1 Extensions for NOM: weighted classes per component and number of classes

A component consists of a group of classes, so it is reasonable that the complexity of the various classes influence the complexity of the resulting component. If the classes are complex, then they are more difficult to understand and maintain, and consequently the component will be complex and difficult to maintain [9]. Using NOM to measure the complexity of the classes, we define the weighted class per component (WCC) as:

$$WCC = \sum_{i=1}^{m} NOM(Ci)$$

3.2 Extensions for DIT: maximum of the DIT and mean of unrelated trees

(Li and Henry, 1993) have evidenced that higher values of DIT identify classes that are hard to maintain: the effort in maintaining group of classes can therefore be indicated by the values of DIT. In particular we initially consider both the highest value of DIT, MAXDIT [9].

$$MAXDIT = \max\{DIT(Ci)\}$$

$$Ci \in k$$

3.3 Extensions for NOC: number of children for component

NOC is extended by counting the number of children of all the classes in the component; we call this metric number of children for a component (NOCC).

$$NOCC = \sum_{i=1}^{m} NOC(Ci)$$

It appears that NOCC is an indicator of reuse inside the component.

3.4 Extensions for CBO: external CBO

We define the level of coupling for a component: external CBO (EXTCBO) as the number of external classes coupled to it.

$$EXTCBO = \sum_{i=1}^{m} ei$$

Where ei is the number of external classes coupled to the class Ci. Clearly EXTCBO evidences the level of dependency of *K* on external entities [9].

3.5 Extensions for RFC: response set for a component

The response set of a component (RFCOM) is the number of all the methods in the member classes and the methods called by those classes. This value is the sum of the values of RFC for all the classes in the set

$$RFCOM = \sum_{i=1}^{m} RFC(Ci)$$

4. QUALITY ASSESSED METRIC OF CBS

Definition of RFCOM metric leads to fact that:

RFCOM = Number of local methods + Number of invoked methods

= NOL + NOC

If all method complexities in a class are considered to be unity, the WCC=n, where n is the no. of methods in the class implies NOL=WCC. By definition of EXTCBO metric; the objects are accessed internally through remote method call (NOC) to other object or instance variable that is:

EXTCBO= Number of invoked method

NOC = EXTCBO

Substituting the metrics WCC and CBO for NOL and NOC respectively in equation (1) then

RFCOM = WCC + EXTCBO…….... (1)

Equation (2) shall be used to identify low quality software.

On examining equation:

RFCOM ≅ WCC + EXTCBO…… (2)

If WCC and EXTCBO both increase then RFCOM also increase, and if WCC and EXTCBO decrease then RFCOM also decrease.

In case of high quality code , component a system are loosely coupled then EXTCBO metric will be low and increase in RFCOM is due to WCC that is including more methods in class increase the RFCOM metric but since coupling is basic, the EXTCBO metric should not significantly increase. Equation RFCOM= NOL + NOC approaches to RFCOM=WCC (i. e. NOL + NOC → WCC).

In case of low quality code, component a system are highly coupled then the CBO metric will be high and increase in RFCOM would be both CBO and WCC. Equation RFCOM=NOL + NOC approach to RFCOM= WCC + EXTCBO (i.e. NOL +NOC → WCC + EXTCBO).

5. METRICS BEHAVIOR UNDER THE CRITERIA COMPLEXITY, EFFORT, RESUABILITY, COUPLING

- WCC value increase then complexity increase.
- MAXDIT value increase then effort increase.
- NOCC value increase then resuability increase.
- EXTCBO value increase then coupling increase.
- RFCOM value increase then coupling increase.

Metrics	Value	Complexity	Effort	Resuability	Coupling
WCC	increase	increase			
MAXDIT	increase		increase		
NOCC	increase			increase	
EXTCBO	increase				increase
RFCOM	increase				ncrease

Table1. Metrics behavior under the criteria complexity, effort, resuability, coupling.

6. EMPIRICAL STUDY

This section describe the five executed code segments, OO metrics and component metrics.

6.1 Program 1 : Multilevel Inheritance

This program implement the multilevel inheritance concept in Object Oriented design and Table 2. show the values of CK Metrics and COTS Metrics for class diagram shown in figure1.

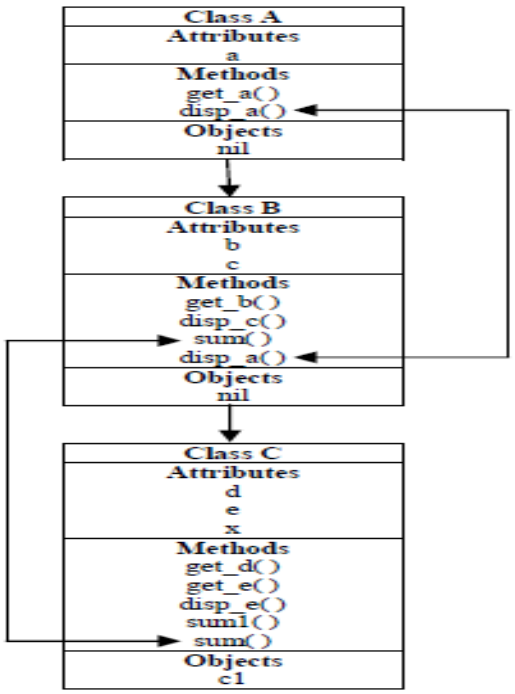

Figure1. Object oriented design multilevel inheritance.

CK METRICS →	WMC	DIT	NOC	CBO	RFC
CLASS A	2	0	2	1	2
CLASS B	4	1	1	2	6
CLASS C	5	2	0	1	11
COTS METRICS →	WCC= 11	MAXDIT = 2	NOCC = 3	EXTCBO = 4	RFCOM = 19

Table2. Value of CK metrics and COTS Metrics for multilevel inheritance.

Value of COTS Metrics for multilevel inheritanc

$$WCC = \sum_{i=1}^{m} NOM(Ci) \quad \text{..... (1)}$$

$$= 11$$

$$MAXDIT = \max_{Ci \in k}\{DIT(Ci)\} \quad \text{..... (2)}$$

$$= 2$$

$$NOCC = \sum_{i=1}^{m} NOC(Ci) \quad \text{..... (3)}$$

$$= 3$$

$$EXTCBO = \sum_{i=1}^{m} ei \quad \text{..... (4)}$$

$$= 4$$

$$RFCOM = \sum_{i=1}^{m} RFC(Ci) \quad \text{..... (5)}$$

$$= 19$$

Calculate the Value of RFCOM

RFCOM $\cong$ WCC+EXTCBO

= 11+4

= 15

6.2 Program 2 : Multiple Inheritance

This program implement the multiple inheritance concept in Object Oriented design and Table 3.show the values of CK Metrics and COTS Metrics for class diagram in figure 2.

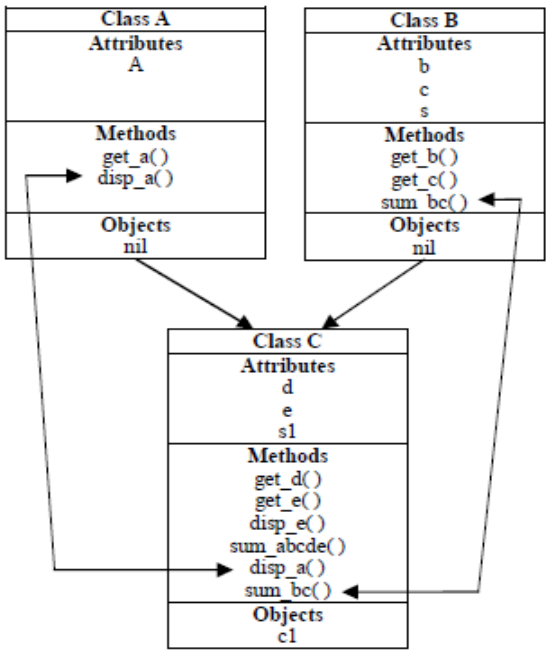

Figure2. Object oriented design multiple inheritance.

CK METRICS →	WMC	DIT	NOC	CBO	RFC
CLASS A	2	0	1	1	2
CLASS B	3	0	1	1	3
CLASS C	6	2	0	2	11
COTS METRICS →	WCC= 11	MAXDIT = 2	NOCC = 2	EXTCBO = 4	RFCOM = 16

Table3. Value of CK metrics and COTS Metrics for multiple inheritance.

Value of COTS Metrics multiple inheritance.

$$WCC = \sum_{i=1}^{m} NOM(Ci) \quad \ldots\ldots\ldots (1)$$

$$= 11$$

$$MAXDIT = \max_{Ci \in k}\{DIT(Ci)\} \quad \ldots\ldots\ldots\ldots (2)$$

$$= 2$$

$$NOCC = \sum_{i=1}^{m} NOC(Ci) \quad \ldots\ldots\ldots (3)$$

$$= 2$$

$$EXTCBO = \sum_{i=1}^{m} ei \quad \ldots\ldots\ldots\ldots\ldots (4)$$

$$= 4$$

$$RFCOM = \sum_{i=1}^{m} RFC(Ci) \quad \ldots\ldots\ldots (5)$$

$$= 16$$

Calculate the Value of RFCOM

RFCOM $\cong$ WCC+EXTCBO

= 11+4

= 15

6.3 Program 3 : Hierarchal Inheritance

This program implement the Hierarchal inheritance concept in Object Oriented design and Table 4. show the values of CK Metrics and COTS Metrics for class diagram shown in figure3.

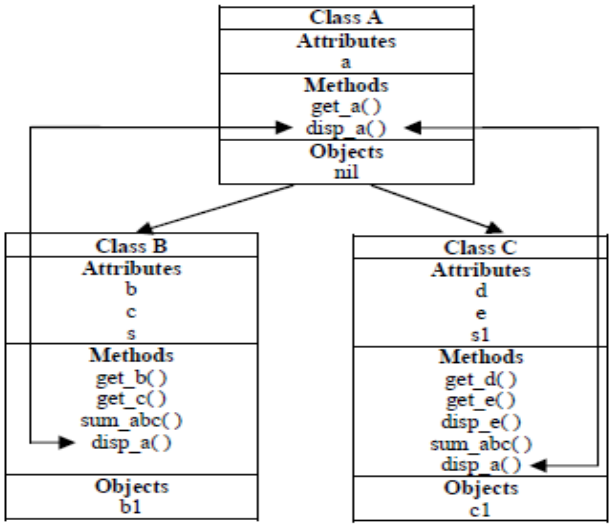

Figure3. Object Oriented design for Hierarchal inheritance.

CK METRICS →	WMC	DIT	NOC	CBO	RFC
CLASS A	2	0	2	1	2
CLASS B	4	1	0	1	6
CLASS C	5	1	0	1	7
COTS METRICS →	WCC =11	MAXDIT = 1	NOCC = 2	EXTCBO = 3	RFCOM = 15

Table4. Value of CK metrics and COTS Metrics for Hierarchal inheritance.

Value of COTS Metrics for Hierarchal inheritance.

$$WCC = \sum_{i=1}^{m} NOM(Ci) \quad \ldots\ldots\ldots (1)$$

$$= 11$$

$$MAXDIT = \max_{Ci \in k}\{DIT(Ci)\} \quad \ldots\ldots\ldots\ldots (2)$$

$$= 1$$

$$NOCC = \sum_{i=1}^{m} NOC(Ci) \quad \text{..... (3)}$$

= 2

$$EXTCBO = \sum_{i=1}^{m} ei \quad \text{..... ..... (4)}$$

= 3

$$RFCOM = \sum_{i=1}^{m} RFC(Ci) \quad \text{..... (5)}$$

= 15

Calculate the Value of RFCOM

RFCOM $\cong$ WCC+EXTCBO

= 11+3

= 14

6.4 Program 4 : Vehicle Classification

This program implement the vehicle classification program in Object Oriented design and Table 5. show the values of CK Metrics and COTS Metrics for class diagram shown in figure 4.

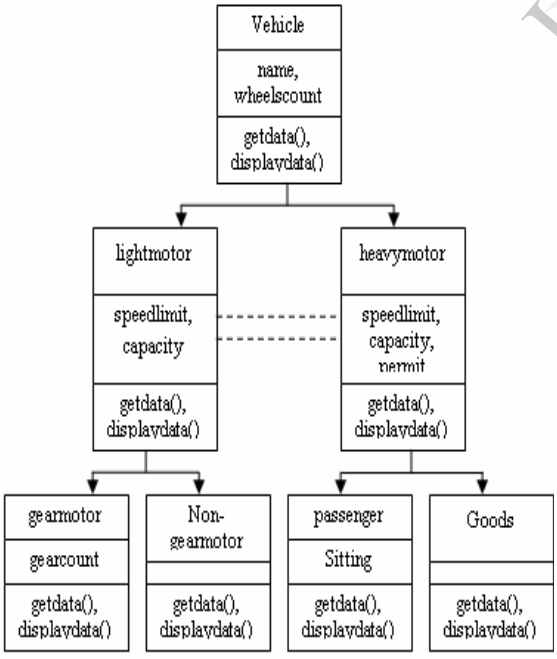

Figure4. Object Oriented design for vehicle classification program.

CK METRICS →	WMC	DIT	NOC	CBO	RFC
Vehicle	2	0	2	0	8
Light Motor	2	1	2	2	6
Heavy Motor	2	1	2	2	6
Gear Motor	2	2	0	0	2
Non-Gear Motor	2	2	0	0	2
Passenger	2	2	0	0	2
Goods	2	2	0	0	2
COTS METRICS →	WCC= 14	MAXDIT = 2	NOCC = 6	EXTCBO = 4	RFCOM = 28

Table5. Value of CK metrics and COTS Metrics for vehicle classification.

Value of COTS Metrics for vehicle classification.

$$WCC = \sum_{i=1}^{m} NOM(Ci) \quad \text{..... (1)}$$

= 14

$$MAXDIT = \max_{Ci \in k}\{DIT(Ci)\} \quad \text{..... (2)}$$

= 2

$$NOCC = \sum_{i=1}^{m} NOC(Ci) \quad \text{..... (3)}$$

= 6

$$EXTCBO = \sum_{i=1}^{m} ei \quad \text{..... ..... (4)}$$

= 4

$$RFCOM = \sum_{i=1}^{m} RFC(Ci) \quad \text{..... (5)}$$

= 28

Calculate the Value of RFCOM

RFCOM $\cong$ WCC+EXTCBO

= 14+4

= 18

6.5 Program 5 : Shapes Drawing

This program implement Shapes drawing program in Object Oriented design and Table6.show the values of CK Metrics and COTS Metrics for class diagram shown in figure5.

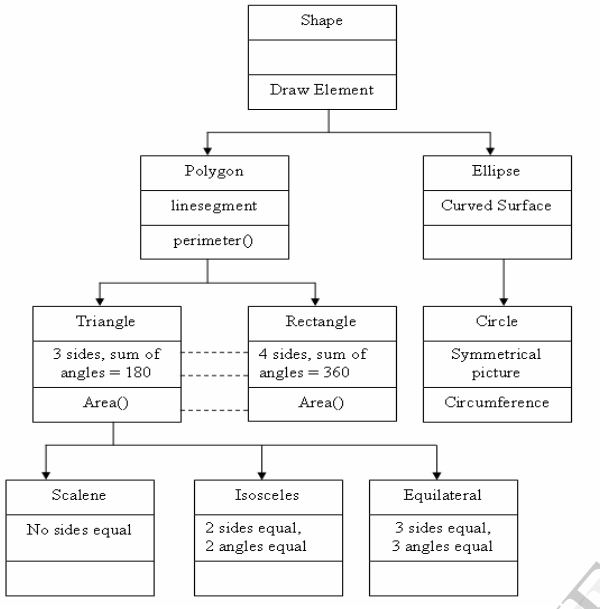

Figure5. Object Oriented design for Shapes drawing program.

CK METRICS →	WMC	DIT	NOC	CBO	RFC
Shape	1	0	2	0	8
Polygon	1	1	2	0	9
Ellipse	0	1	1	0	0
Triangle	1	2	3	3	4
Rectangle	1	2	0	3	4
Circle	1	2	0	0	1
Scalence	0	3	0	0	0
Isosceles	0	3	0	0	0
Equilateral	0	3	0	0	0
COTS METRICS →	WCC= 5	MAXDIT = 3	NOCC = 8	EXTCBO = 6	RFCOM = 26

Table6. Value of CK Metrics and COTS Metrics for Shape drawing.

Value of COTS Metrics for Shape drawing

$$WCC = \sum_{i=1}^{m} NOM(Ci) \quad \ldots\ldots\ldots (1)$$

= 5

$$MAXDIT = \max_{Ci \in k}\{DIT(Ci)\} \quad \ldots\ldots\ldots\ldots (2)$$

= 3

$$NOCC = \sum_{i=1}^{m} NOC(Ci) \quad \ldots\ldots\ldots (3)$$

= 8

$$EXTCBO = \sum_{i=1}^{m} ei \quad \ldots\ldots\ldots\ldots\ldots (4)$$

= 6

$$RFCOM = \sum_{i=1}^{m} RFC(Ci) \quad \ldots\ldots\ldots (5)$$

= 26

Calculate the Value of RFCOM

RFCOM $\cong$ WCC+EXTCBO

= 5+6

= 11

7.RESULT ANALYSIS

In case of loosely coupled code EXT CBO will be low and value of RFCOM will less. This implies quality of code will be high. Program 5 has highest quality than other program. Thus once we have calculated the values of the metrics then quality level can be measured.

$$\text{Quality level } \alpha \frac{1}{\text{RFCOM}}$$

highest

PROGRAM	RFCOM=WCC+EXTCBO
PROGRAM 5	11
PROGRAM 3	14
PROGRAM 1	15
PROGRAM 2	15
PROGRAM 4	18

lowest

Table7. RFCOM of quality level value of Various programs.

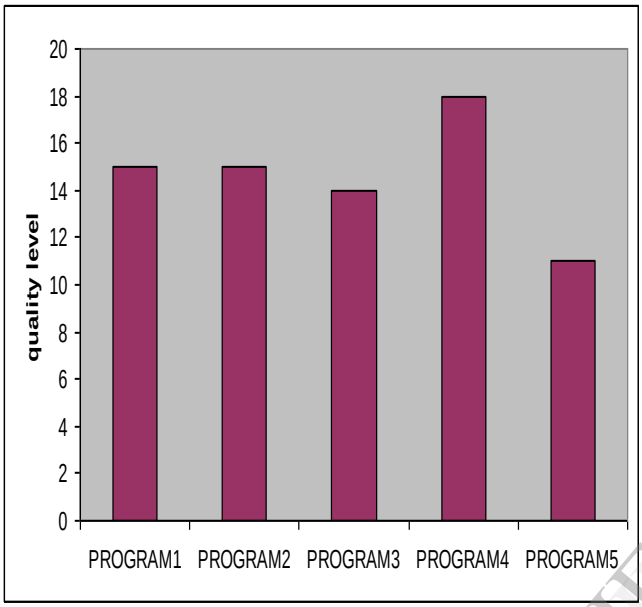

Figure6. Comparision chart of quality level of various programs.

8. CONCLUSION

In this paper, we have introduced new COTS Metrics from corresponding CK Metrics and a model the measure the quality of component software design. We have proposed model based on the Component based Metrics (RFCOM, WCC, EXTCBO) derived for CK Metrics (RFC, WCM, CBO). This model help us such to quality of component based software design.

REFRENCES

[1] V. Lakshmi Narasimhan, P. T. Parthasarathy and M. Das, "Evaluation of a suite for Component Based Software Engineering", vol. 6, pp.731-740, 2009.

[2] Ivica Crnkovic, "Component-based Software Engineering- New Challenge in Software Development", Proceedings of the 25th International conference on Information Technology interface, pp. 9-18, September 2003.

[3] Gurdev Singh, Dilbag Singh and Vikram Singh, "A study of Software Metrics" International Journal of Computational Engineering and Management, vol. 11, pp. 476-492, January 2011.

[4] Parul Gandhi, Pradeep Kumar Bhatia, "Reusability Metrics for Object Oriented System", An Alternative Approach International Journal of Software Engineering (IJSE), vol. 1, Issue 4, 2002.

[5] Shyam R. Chidamber and Chris F. Kemerer, "A Metric suite of Object Oriented Design", IEEE Transaction on Software Engineering, vol. 20, No. 6, pp. 476-493, 1994.

[6] Chidamber and Kemerer, "A suite of Metrics", Japan Advanced Institute of science and Technology School of Information Science, May 2008.

[7] Pradeep Bhatia, Yogesh Singh and H. L. Verma, "An Empirical Study for Assessing Quality of OO Code", vol. 14(3), pp. 385-400, 2002.

[8] Shyam R. Chidamber , Chris F. Kemerer , "A Metric suite for Object Oriented design", M.I.T. Solan School of Management E53-315, 1993.

[9] Giancarlo Succi, Luigi Benedicenti and Martin Mintchev, "Defining Metrics for Software Components", Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, vol. 10, pp. 16-23, 2000.

[10] Washizaki, Yamamoto, Fukazawa, "A Metrics Suite for Measuring Reusability of Software Components", Proceedings of the ninth international Conference on Software Engineering, pp. 211-213, 2003 .

[11] Briand, L. C., Daly, J. W., and Wust, J. K., "A Unified Framework for Coupling Measurement in Object-Oriented Systems," IEEE Transactions on Software Engineering, vol. 25, pp. 91-121, January/February 1999.