ISSN: 0976-3031

Available Online at http://www.recentscientific.com

CODEN: IJRSFP (USA)

International Journal of Recent Scientific Research
Vol. 8, Issue, 7, pp. 18148-18154, July, 2017

International Journal of Recent Scientific Research

DOI: 10.24327/IJRSR

Research Article

IMPLEMENTATION AND EVALUATION OF COTS METRICS

Surbhi Goyal[1] and Pradeep Kumar Bhatia[2]

[1,2]Department of CSE, G. J. University of Science & Technology, Hisar, Haryana, India

DOI: http://dx.doi.org/10.24327/ijrsr.2017.0807.0460

ARTICLE INFO

Article History:

Received 15th April, 2017
Received in revised form 25th May, 2017
Accepted 28th June, 2017
Published online 28th July, 2017

Key Words:

COTS, Reusability, Adaptability, Complexity.

ABSTRACT

Now a days it is necessary to measure the reusability of a component as this is the most effiective way to increase our productivity. Building new software from pre-existing software is Component Based Software Engineering. COTS stands for Commercial off-the shelf products. COTS can be used for building new software from pre-existing components. In this paper, we introduce new metrics for COTS so that we can measure the quality of a software product as this is very useful for building new software now a days. We validate our study by Java programming examples and calculate CK metrics for COTS based on object oriented metrics and also define some metrics based on reusability.

INTRODUCTION

Software engineers should apply effective metrics with new tools within the context of a mature software process to evaluate reusable suite of software system. [12] A software component is a defined package of software implementation which offers well defined functionality and can be reuse for building new application. Reusability increases efficiency and decreases cost, time, effort etc.

Commercial off-the shelf (COTS) are the ready- made products which can be used as "it is". These products are designed to be easily installed and to interoperate with existing system components.

As people develop components, there is a need for defining suitable metrics to measure such components. Software metric is a measure of degree to which software system or process possesses some property. Metrics are very useful to measure the quality of software. If we know the quality of software then we can reuse the software. Basically COTS are reusable products and we can reuse them if we know the quality like cohesion, coupling, reuse factor etc. The goal of this work is on definition and validation of metrics for components. The Chidamber and Kermerer (CK) metrics suite is a "de-facto" standard for measuring properties of classes and objects. We base our work on an extension of such metrics In addition, (Briand *et al.*,1996) has set reference properties that size, length, complexity, coupling and cohesion measure must follow: the proposed metrics follows such properties. We also define some new metrics for reusability.

Component Based Software Engineering aims to build software from pre-existing components, build components as reusable entities and evolve application by replacing component. Now days reusing any product while building any new software are very common. While reusing any software it is necessary to measure the reusability of the product effectively because reusability is an effective way to improve productivity. But if we want to reuse any product we can face many problems like it can increase the complexity of the new software we are building, it can increase the cost, quality of the pre-existing software is not good for the new software etc. Therefore we should define some measure or quality which can address this problem. These measures can be defined by metrics which is a standard of measure of a degree to which a software system or process possesses some property.

Object Oriented Ck Metrics

Object oriented CK metrics are given below:

Weighted Methods per Class (WMC)

The WMC is a count of the methods implemented within a class or the sum of the complexities of the methods [2]. This metric is used to measure the understandability, reusability and

Corresponding author:* **Surbhi Goyal
Department of CSE, G. J. University of Science & Technology, Hisar, Haryana, India

maintainability.WMC is the predictor of how much Time and Effort is required to develop and to maintain the class. Greater the number of methods, more is the impact on the children. Classes with large WMC are likely to have more faults, limiting the possibility of re-use and making the effort expended one-shot investment. Large WMC increases the density of bugs and decreases the quality of software [7].

Depth of Inheritance Tree (DIT)

DIT is defined as the maximum length inheritance path from the class to the root class. Classes with large DIT are likely to inherit, making more complex to predict its behavior. Greater value of DIT leads to greater the potential re-use of inherited methods. Large DIT increases density of bugs and decreases the quality of software. Small values of DIT in most of the system's classes may be an indicator that designers are for asking re-usability for simplicity of understanding [1,6,7].

Number Of Children (NOC)

NOC is defined as the number of immediate subclasses subordinated to a class in the class hierarchy. Greater NOC leads to greater re-use, probability of improper abstraction of parent class. Large NOC leads to more testing and misuse of sub-classing. Large NOC leads to poor design and high complexity. High NOC leads to high reuse which indeed less faults. Small values of NOC may be an indicator of lack of communication between different class designers [6,7].

Coupling between Objects (CBO)

Coupling is a measure of strength of association established by a connection from one entity to another. 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[2]. It is measured the counting the distinct non inheritance related class hierarchy on which a class depends. Small value of CBO improves modularity and promotes encapsulation. The larger the number of couples, the higher the sensitivity to changes in other parts of the design and therefore maintenance is more difficult. Small CBO indicates independence in the class, making easier to re-use and also makes easier to test a class.

Response For Class (RFC)

RFC is defined as the number of methods of the class plus the number of methods called by any of those methods. The response set of a class is a set of methods that can potentially be executed in response to a message received by an object of that class. RFC is simply the number of methods in the set.

If a large numbers of methods are invoked from a class testing, debugging and maintenance of the class becomes more complex and it becomes hard to understand. High RFC leads to more fault-proneness. High RFC increases density of bugs and thereby decreases the quality [3,7].

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. The number of pairs of member functions without shared instance variables, minus the number of pairs of member functions with shared instance variables. However, the metric is set to zero whenever this subtraction is negative [2,4,5]. High cohesion indicates good class subdivision. Lack of Cohesion or low cohesion increases complexity, thereby increasing the likelihood of errors during the development process. It does not promote encapsulation and implies classes should probably be split into two or more subclasses. Indicates the low quality design of the software.

Part 1: Cots Metrics from Ck Metrics

Weighted Class per Component (WCC)

It is an extension for NOM. .Complexity of different classes of a COTS product affect the complexity of a resulting COTS product. If the classes are complex then the product will be complex and that product will be more difficult to understand and maintain. Therefore we can define weighted class per component as:

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

Maximum Depth of Inheritance (MAXDIT)

It is an extension for DIT. If the DIT increases then effort increases. MAX of DIT can be calculated as:

MAXDIT=max{DIT(Ci)}
Ci € k

Number of Children for Component (NOCC)

It is as an extension of NOC. NOCC can be calculated as the sum of number of children of all the classes in the component.

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

External Coupling Between bjects (EXTCBO)

EXTCBO can be calculated as given below

$$\text{EXTCBO}= \sum_{i=1}^{m} (e_i)$$

Where ei is the number of external classes coupled
With the class Ci.

Response Set for a Component.(RFCOM)

It an extension for RFC. This is the number of all the methods in the member classes and the methods called by those classes. It can be calculated as:

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

RFC= No. of local methods + No. of invoked methods
RFC =NOL+NOC

When all the methods complexity in the class are imagined to be unity then NOL=WMC and CBO= No. of invoked methods, NOC=CBO

Therefore
RFC=WMC+CBO
In terms of COTS Metrics
∑RFC=∑WMC +∑CB0
RFCOM= WCC + EXTCBO

For *high* quality code components, components are *loosely* coupled then EXTCBO metric will be *low* and increase in RFCOM is due to WCC. Therefore, RFCOM = WCC + EXTCBO approaches to RFCOM=WCC.

For *low* quality code components, components are *highly* coupled then EXTCBO metric will be *high* and increase in RFCOM is due to WCC as well EXTCBO. Therefore, RFCOM= WCC + EXTCBO

Part 2: Cots Metrics Based on Reusability

Not only above metrics are useful for future reuse, reusability metrics are one of the important concern while reusing any component. Therefore, we define reusability metrics given below:

Reusability: Reusability is one of the most important aspect of software component. It is used to measure the degree of one component that can be reused [8]. Some aspects used for reusability:

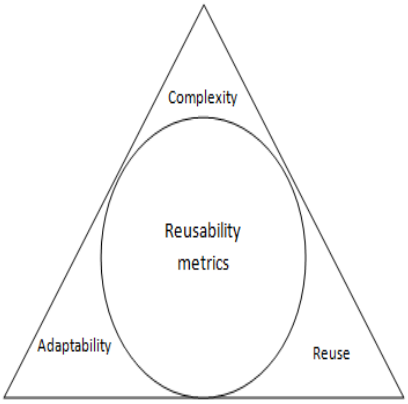

Fig 1 Reusability Factors

- ***Availability:*** It determines how easy and fast is to retrieve a software component.
- ***Quality:*** It is one of the important aspects while reusing any component. It is regarded as a characteristic which describes how good it fulfills its requirements and also how error and bug free a component is [8].
- ***Adaptability:*** Adaptability of a component is defined as a metric that how good a component is to adapt in a different environment [8]. The adaptability of a component can be written as Rate of Component Adaptability (RCA). RCA depends on method complexity (MC) and interface complexity(IC) [8]. MC is the method dependency and can be calculated as:

$$MC = \frac{\sum_{i=1}^{n} CC_i}{n}$$

Where
∑CCi= sum of cyclomatic complexity of methods.
n = total numbers of methods in a class.

Cyclomatic complexity=E-N+X
Where E is number of edges, N is number of nodes or decision points and X is number of exists.

Interface complexity gives the source of information to understand and reuse the component. IC should be as low as possible and it can be written as component interaction density (CID). CID metric measures the ratio of actual number of interactions to the available number of interaction in component [8].

$$CID = \frac{\text{Number of actual interaction}}{\text{Maximum available interactions}}$$

Reuse: The actual reuse of a component can also be used to infer how usable and how easy it is to adapt it. The amount and frequency of reuse, especially in contexts similar to that of the developer can serve as reference points and she or he may select the component determines how expensive it is to reuse. Reuse can be expressed by class reusability and method reusability. We have to give class and method ranks for these calculations.

$$\text{Class Reusability}(f(Ci) = \frac{r(Ci)}{\sum_{k=1}^{n} Ck}$$

where
r(Ci) is the sum of all classes ranks,c is a class that is used by the classes c1….cn

$$\text{Method Reusability}(f(Mi) = \frac{r(Mi)}{\sum_{k=1}^{m} Mk}$$

where
r(Mi) is the sum of all method ranks, m is a method that is used by the methods m_1….m_n.

Complexity: complexity is one of the most important factors which are considered while reusing any component again. There are three co-factors for calculating complexity which are:

Cyclomatic complexity: This is used after the implementation of component if finished and cyclomatic complexity of component (CCC) is given as:

$$CCC = \sum_{i=1}^{m} CCi + DIM$$

DIM= depth of inherited method inside the class.

Coupling: Coupling defines interdependency. In this, we count the way in which one module depend on other. In general, coupling should be as low as possible. We calculate component coupling average (CCA) which is given as:

$$CCA_{coup} = \frac{MCFC}{TC}$$

Where, MCFC= method complexity based on class
TC= total no. of class in component.

Cohesion: Cohesion describes the similarity of method between classes. In general, cohesion should be as high as

possible. We calculate Component Cohesion Average (CCA_{coh}) which is given as:

$$CCA_{coh}= \begin{cases} \frac{TCCh}{TC} & TCCh>0 \\ 0, & \text{otherwise} \end{cases}$$

Where, TCCh= total class cohesion based on methods which is given by the max. no. of similarity of method situation in class.

Empirical Study

This section is divided in two parts. Part 1: It describe the four executed code segments, OO metrics and COTS metrics, part 2: it describes reusability metrics for COTS.

Part 1

Program 1: Multilevel inheritance: This program implement the multilevel inheritance concept in Java design and Table 2. Show the value of CK metrics and COTS metrics for class diagram shown below:

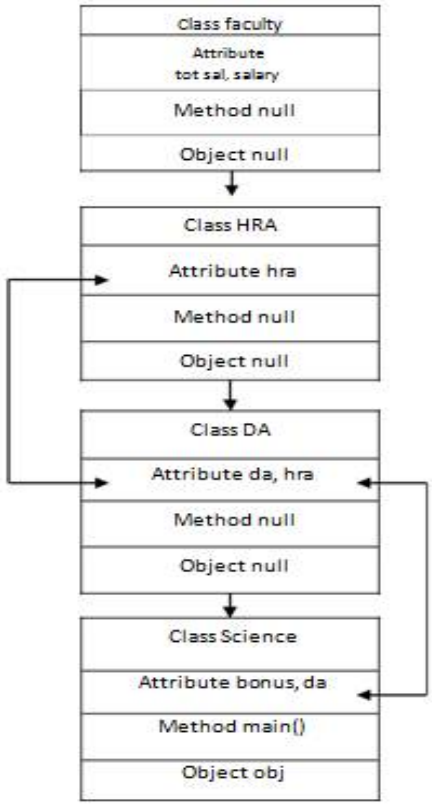

Fig 1 class diagram for multilevel inheritance

Table 2 Value of CK metrics and COTS metrics for multilevel inheritance

CK Metrics	WMC	DIT	NOC	CBO	RFC
Class faculty	1	0	3	0	1
Class HRA	1	1	2	1	2
Class DA	2	2	1	2	4
Class Science	2	3	0	1	6
COTS Metrics	WCC=6	MAXDIT=3	NOCC=6	EXTCBO=4	RFCOM=13

$$RFCOM=\sum_{i=1}^{m} RFC\ (Ci)=13$$

Program 2: Hierarchical Inheritance

This program implements the hierarchical inheritance concept in java design and Table 3. Show the value of CK metrics and COTS metrics for class diagram shown below:

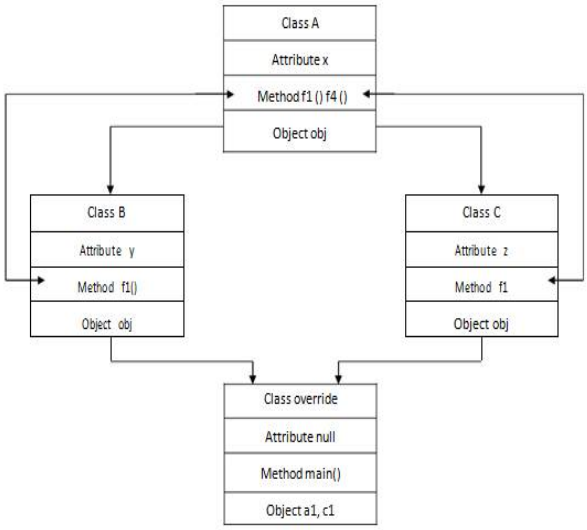

Fig 2 Class diagram for hierarchical inheritance

Table 3 Value of ck metrics and COTS metrics for hierarchical inheritance

CK Metrics	WMC	DIT	NOC	CBO	RFC
Class A	2	0	2	2	2
Class B	1	1	0	1	3
Class C	1	2	0	1	3
COTS Metrics	WCC=4	MAXDIT=2	NOCC=2	EXTCBO=4	RFCOM=8

Value of COTS metrics for hierarchical inheritance

$$WCC=\sum_{i=1}^{m} NOM\ (C_i)=4$$

MAXDIT=max {DIT(Ci)}=2
Ci € k

$$NOCC=\sum_{i=1}^{m} NOC\ (Ci)=2$$

$$EXTCBO=\sum_{i=1}^{m} (e_i)=4$$

$$RFCOM=\sum_{i=1}^{m} RFC\ (Ci)=8$$

Program 3: Person classification: This program implements the person classification concept in java design and Table 4. Show the value of CK metrics and COTS metrics for class diagram shown below:

Value of COTS metrics for person classification

$$WCC=\sum_{i=1}^{m} NOM\ (C_i)=12$$

MAXDIT=max {DIT(Ci)}=2
Ci € k

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

EXTCBO= $\sum_{i=1}^{m} (e_i)$ =4

RFCOM= $\sum_{i=1}^{m} RFC$ (Ci)=24

Table 4 Value of CK Metrics and COTS metrics for person classification

CK Metrics	WMC	DIT	NOC	CBO	RFC
Class Person	4	0	2	1	4
Class Employee	4	1	1	2	8
Class Hourly Employee	4	2	0	1	12
COTS Metrics	WCC=12	MAXDIT=2	NOCC=3	EXTCBO=4	RFCOM=24

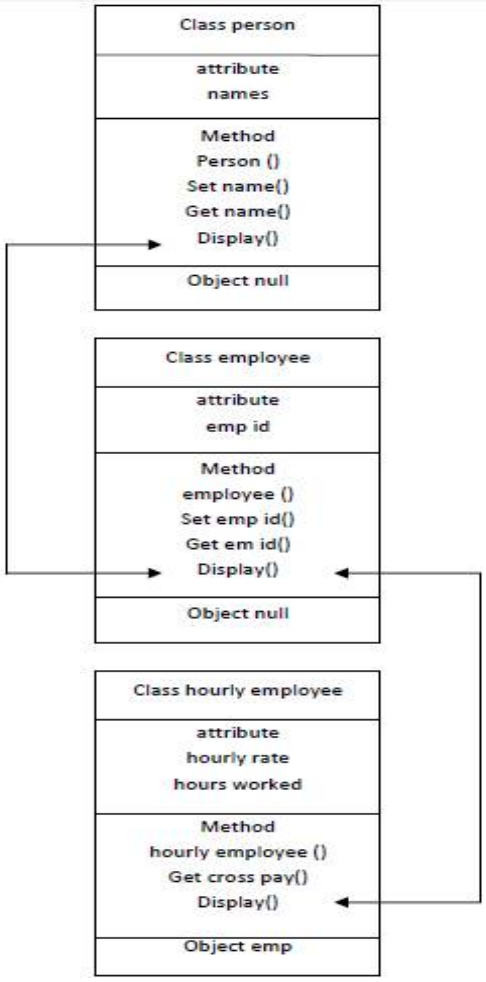

Fig 3 Class diagram for Person Classification

Program 4: Hybrid Inheritance: This program implements the hybrid inheritance concept in java design and Table 5. Show the value of CK metrics and COTS metrics for class diagram shown below:

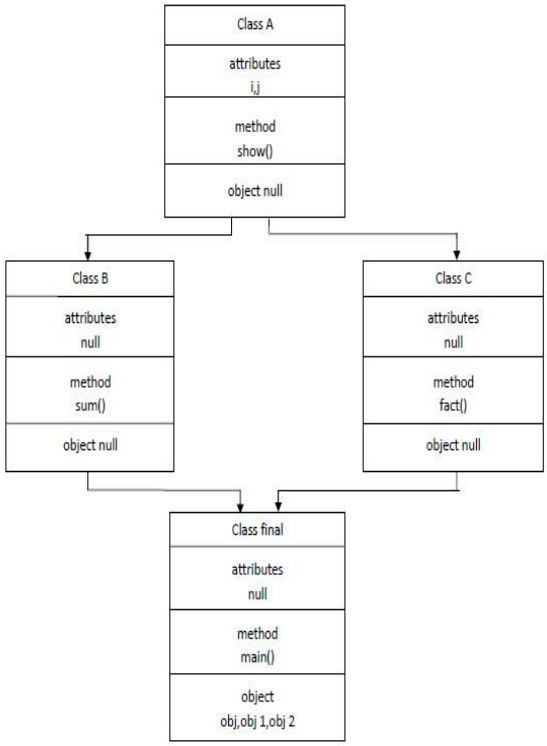

Fig 4 Class diagram for hybrid inheritance

Table 5 Value of CK Metrics and COTS metrics for Hybrid inheritance

CK Metrics	WCC	DIT	NOC	CBO	RFC
Class a	1	0	2	0	1
Class b	1	1	0	0	2
Class c	1	1	1	0	2
Class d	0	2	0	0	3
COTS Metrics	WCC=3	MAXDIT=2	NOCC=3	EXTCBO=0	RFCOM= 8

Value of COTS metrics for Hybrid inheritance

WCC= $\sum_{i=1}^{m} NOM\ (C_i)$=3

MAXDIT=max {DIT(Ci)}=2
Ci € k

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

EXTCBO= $\sum_{i=1}^{m} (e_i)$ =0

RFCOM= $\sum_{i=1}^{m} RFC$ (Ci)=8

Part 2

We calculate adaptability, reuse, and complexity metric for employee classification program whose flow diagram is given below. In this program we have one class and five methods which are ge tName(), getcomName() ,getSalary(), getId(), show All(), end

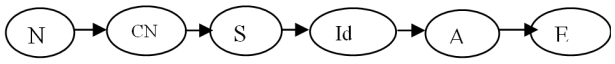

Fig 5 Flow diagram for employee classification
COTS Adaptability

$$MC = \sum_{i=1}^{n} CCi \div n$$

Cyclomatic complexity=5-0+1=6

n=5
MC=6/5=1.2

IC=0, because there is no interaction between classes as there is only one class.

COTS Reuse

As there is only one class therefore rank is one and sum of classes is also one

Therefore,

f(Ci)=1/1=1
As there is six methods therefore rank is six and sum of methods is also six.
Therefore,
f(Mi)=6/6=1

COTS Complexity

$$CCC = \sum_{i=0}^{m} CCi + DIM$$

$$\sum_{i=0}^{m} CCi = 1+1+1+1+1 = 5$$

DIM= 0 (As there is only one class)

COTS Coupling

CCA_{coup} = MCFC/ TC
MCFC = method complexity based on class
= no. of methods in the class= 5

TC= total no. of class in a component= 1
CCA_{coup}= 5/ 1= 5

COTS Cohesion

$$CCA_{coh} = \begin{cases} TCCh/\ TC, & TCCh>0 \\ 0, & \text{otherwise} \end{cases}$$

TCCh= 0 (As there are five different methods in the class)
TC= total no. of classes in a component= 1
Otherwise condition is applicable here
Cohesion= 0

RESULTS

Values of COTS metrics EXTCOM and RFCOM obtained from above four programs ae summarized into tables 6.

Table 4 illustrates that program 4 in Part 1 has the lowest EXTCBO value and RFCOM is also lowest for this program, implying this program has high quality code.

Tables 6 Summary of Values of COTS metrics

	EXTCBO	RFCBO
Program 1	4	13
Program 2	4	8
Program 3	4	24
Program 4	0	8

Part 2: Adaptability and reuse factors are greater than 1 therefore component can be reuse. But there is high coupling and zero cohesion. There is ***less*** cohesion and ***high*** coupling. This shows that this class is ***less*** reusable Therefore; we can say that this component is not good for future reusability. As in this study this has been seen that the continuous increase of the reused component number, in order to develop software, how to choose the component with improved reusability from the component library is a crucial problem for the developers of the component library and the persons of reusing components

CONCLUSION

In this paper we introduced COTS metrics from CK metrics and reusability metrics for COTS so that we can reuse any software again. We conclude that quality of software depend on its RFCOM value, Reusability of software depend adaptability, reuse, and complexity factor. RFCOM value should be low because low value of RFCOM means low value of coupling and complexity factor. Adaptability and reuse factor should be greater than complexity (low coupling and high cohesion) for better use in future.

References

1. Ahmed M. Salem, Abrar A. Qureshi, "Analysis of inconsistencies in object oriented metrics", *Journal of Software Engineering and Applications,* Vol. 4, pp.123-128, January 2011.
2. Sonu Dhariwal, Ashwani Kumar, Pradeep Kumar Bhatia," Design of COTS Metrics based upon CK Metrics", *International Journal of Engineering Research and Technology*, vol.2 Issue 9, September 2013.
3. Pradeep Bhatia, Yogesh singh, H.L. Verma, "An emperical study for accessing quality of OO code", *Ultra Science*, Vol. 14, No.3, pp.385-400, May 2002.
4. Tibor Gyimo thy, Rudolf Ferenc, Istvan Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction", *IEEE Transactions on Software Engineering*, Vol. 31, No. 10, pp.897-910, October 2005.
5. Chidamber Shyam, Kemerer Chris, Darcy David, "Managerial use of metrics for object- oriented software: an exploratory analysis", *IEEE Transactions on software Engineering*, Vol 24, No. 8, pp.629-639, August 2008.
6. http://www.aivosto.com/project/help/pm-oo-ck.html.
7. sdlab.naist.jp/members/camargo/presentations/CKMetrics.ppt
8. Suchita Yadav, Dr. Pradeep Tomar, Sachin Kumar," Metrics Suite for Accessing the Reusability of Component Based Software", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, Issue 5, May 2014.
9. Khaled Musa and Jawad Alkhateeb, "Quality Model

Based On COTS Quality Attributes", *International Journal of Software Engineering and Application*, Vol.4, no.1, January 2013.

10. Macro Torchiano, Letizia Jaccheri, Cari-Fredrik Sorensen and All Inge Wang,"COTS Products characterization", *SEKE* 02, July 15-19, 2002.
11. Sahra Sedigh, Raymond A.Paul, Arif Ghafoor," Software Engineering Metrics for COTS based System", Institute of Electrical and Electronics Engineering (IEEE), May 2001.
12. Puneet Goswami, Pradeep Kumar, OP sangwan, "A Metrics Methodology For Predicting Reusable Suite of Component Based Software System", *International Journal of Computer Science and Security*, vol.4,Issue 1, 2010.

How to cite this article:

Surbhi Goyal and Pradeep Kumar Bhatia.2017, Implementation and Evaluation of cots Metrics. *Int J Recent Sci Res.* 8(7), pp. 18148-18154. DOI: http://dx.doi.org/10.24327/ijrsr.2017.0807.0460

****