A Design of Cohesion and Coupling Metrics for Component based Software Systems

Pooja Rana
Research Scholar
Department of CSA
Maharashi Dayanand University

Rajender Singh
Professor
Department of CSA
Maharashi Dayanand University

ABSTRACT
Component based software engineering (CBSE) is based on the concept of reusability. CBSE is upcoming paradigm where emphasis is laid on reuse of existing component and rebuilds a new component. Software metrics are used to check the complexity of software. Many software metrics have been proposed for CBS to measure various attributes like complexity, cohesion, coupling etc. Many different cohesion and coupling metrics have been developed. For quality software the cohesion should be high and coupling should be low. The aim of this paper is to develop adequate coupling, cohesion and interface metrics. Graph notation and concept of weights have been used to illustrate proposed metrics and evaluate the results accordingly.

Keywords
Component based software engineering (CBSE), Coupling, Cohesion, Interface Metrics.

1. INTRODUCTION
Software components are prefabricated building blocks that perform specific functions and that can communicate with each other using industry standard messaging interfaces. Distinct from software objects, components are larger modules that represent a higher level of functionality. A component is something that can be deployed as a black box. It has an external specification, which is independent of its internal mechanisms. Component based software engineering (CBSE) denotes the process of building software by using pre-built software components thus basing on the meaning of software components.

Metrics and Measurements are a key element for controlling software engineering process. Software metrics are quantifiable measures that could be used to measure different characteristics of a software system or the software development process. Software metrics play a very important role in assessing and predicting various attributes of software such as complexity, reusability, maintainability, testability etc. Among these attributes complexity affects all other attributes of the software. Software metrics are essential to plan, predict, monitor, control, evaluate, products and processes. The main goal of the software metrics is to reduce costs, Improve quality, Control/ Monitor schedule, small testing effort, many reusable fragments, to better understand the quality of the product and the program. The paper is organized in different sections; section 2 describes literature review of some basic coupling and cohesion metrics. Section 3 describes the proposed cohesion and coupling metrics, section 4 represents the empirical evaluation of the metrics. In the last paper concludes with a discussion of the implications of the research.

2. TRADITIONAL COUPLING AND COHESION METRICS
Coupling is a measure of the degree of independence between modules. When there is little interaction between two modules, the modules are described as loosely coupled. When there is a high degree of interaction the modules are described as tightly coupled. The component complexity closely depends on what contributes to develop components. There are many factors that affect the component Complexity like variables, interface, coupling and cohesion cyclometic complexity. Variable factors define the complexity of the variables in the component. Interface means the interaction of one component with other component. Coupling is interdependence between the components. Cohesion is interdependence of variables and methods of a component. Last factor is cyclometic complexity of the methods of the component.

Table 1 summary of complexity metrics

Name	Definition
CDM[13]	“The complexity results from dependencies among system’s components. Dependency of a component C_i to other component is the number of all paths in the graph from C_i to other component.”[13]
CIDM[13]	“This Metric computes the ratio of total number of direct interactions between the components to total number components.”[13]
TC(CBS)[9]	This composite metric takes different attributes of complexity. “The result shows the effect of these parameters on complexity of a CBS.”[9]
IACC[9]	This metric shows the interaction with other component. The concept of link is used to quantify interface aspect of a component.[9]
AIIC[6]	This metric shows the average of the incoming interactions of one component
AOIC[6]	This metric shows the average of the outgoing interactions of one component
AIC(CBS)[6]	This Metric shows the average interface metric by summation of incoming interface and outgoing interface metrics.

Table 1 summaries different type of the complexity metrics. The dependency among components may be defined as the reliance of a component on others to support a specific functionality or configuration [9]. In CBSE system the components interacts with other components by sharing information in order to provide system functionalities. This composition creates interaction that promotes dependencies among components. System functionalities cannot solely encapsulate within one component. Therefore changing a component may affect that composite functionality, which is reflected in different components. In addition, replacing a new version of a specific component might involve replacing the component on which it depends, in order to preserve a specific system‘s functionality.

Cohesion is the measure of strength of the association of elements within a module. In other words, the extent to which all instructions in a module relate to a single function is called cohesion. Table 2 summarizes the characteristics of the cohesion metrics.

Table 2 summary of the cohesion metrics

Name	Definition
LCOM[19]	"This metric calculate the number of pairs of methods in class using no instance variable in common." [19]
LCOM3[18]	"Number of disjoint components in the graph that represents each method as a node and the sharing of at least one attribute as an edge." [18]
RLCOM[15]	"Ratio of number of non-similar method pairs to total number of method pairs in the class".[15]
TCC[17]	"Ratio of number of similar method pairs to total number of method pairs in class." [15]

These cohesion metrics considered method similarity as an intransitive relation. LCOM3 and TCC incorporate indirect relationships between methods. LCOM3 and TCC treat indirect and direct cohesion in the same way[14].

3. PROPOSED COHESION AND COUPLING METRICS

3.1 Cohesion Metrics

Cohesion is the measure of strength of the association of elements within a component. In a truly cohesive component, all of the instructions in the component pertains to performing a single unified task. The cohesive component only needs to take the data it is passed, act on them, and pass its output on to its super-ordinate component. Cohesion specifies the similarity of methods in a component. It is a measure of the extent to which the various functions performed by a component are related to one another.

COVC (Cohesion of variables within a component):

Cohesion of variables in a component refers to the frequency of variables usage by the component. A component is cohesive if the association of variables declared in the component is focused on accomplishing a single task. The instance variables are classified in three categories standard, moderate and Critical. This classification is based on data types of the instance variables. Standard include integer, float, double, Boolean etc., moderate includes string, arrays, vector, list, Critical includes class type, user defined component, pointers and references. Suppose a component C such as a class has a set of methods $M(C)=\{m_{c1}, m_{c2}, m_{c3}, \ldots\ldots\ldots, m_{cn}\}$ and a set of instance variables v in $V(C) = \{v_{c1}, v_{c2}, v_{c3}, \ldots\ldots\ldots V_{cn}\}$. Fv(C) is the set of pairs (v_c, m_c) for each instance variables v in V(C) that is used by methods m in M(C). Fv(C) is further divided into three i.e. a set of pairs (v_{si}, m_{ci}) and a set of pairs (v_{mi}, m_{ci}) and a set of pairs(v_{ci}, m_{ci}) for each instance variable v in V(C) that is used by methods m in M(C).

$$COVC=\sum_{i=0}^{n}\frac{FIV}{TV}$$

$$FIV=\sum_{i=0}^{n}\{[f(vsi)*Ws]+[f(vmi)*Wm]+[f(vci)*Wc]\}$$

Here

FIV = frequency of the instance variables within a component

TV= total no of Instance Variable in a component

$F(v_{si})$= Frequency of standard variables

$F(v_{mi})$= Frequency of moderate variables

$F(v_{ci})$= Frequency of critical variables

Ws, Wm, Wc are the weight factor of the standard, moderate and critical type of variables respectively.

COMC (Cohesion of Methods within a component):

Cohesion of Methods in a component refers to the relatedness of methods and instance variables of a component. This metrics considers the interaction between the methods with in a component. Here we find out the sum of methods that use the same type of variables i.e standard, moderate, critical.

$$COMC=\sum_{i=0}^{n}\frac{COM}{TM}$$

$$COM=\sum_{i=0}^{n}\{(Msi*Ws)+(Mmi*Wm)+(MCi*Wc)\}$$

COM = count of methods that use same type of variables

TM= total no of methods

Ms_i= sum of methods that use Standard type of variables.

Mm_i= sum of methods that use Moderate type of variables.

Mc_i= sum of methods that use critical type of variables.

Ws, Wm, Wc are weight factor for standard, moderate and critical type of variables.

TCCC(Total Cohesion Complexity of a Component)

Total cohesion complexity of a component is the combination of cohesion of variables in a component and cohesion of methods in a component.

TCCC= COVC + COMC

COVC= cohesion of variables within a component matrices

COMC= cohesion of methods within a component matrices

3.2 Coupling Metrics

Coupling between components is the number of other components coupled to this component. In CBSS, coupling will be defined as: two components are coupled if and only if at least one of them acts upon other. Coupling and cohesion relate to particular relationships that exist between component and within component respectively.

In order to develop a coupling metrics a directed graph(G) is to be taken into consideration. The vertices of a graph are components and the edges between the vertices are interface between the components. From this directed graph an interface matrix(IM(n*n)) is derived. In this matrix one represents the interface between the component and zero represents that there is no interface among the component.

$$IM[I,j] = \begin{cases} 1 \; if\ there\ is\ an\ interface\ between\ the\ component\ Ci\ and\ Cj \\ 0 \; if\ there\ is\ no\ interface\ between\ the\ component\ Ci\ and\ Cj \end{cases}$$

Suppose there is a set of components c={c_1, c_2, c_3 ….. c_n}. let there is a set of IN parameters and OT parameters related to each component. These parameters are further classified into three categories standard, moderate and critical. Each return value is considered as IN parameter and arguments passed as OUT parameters.

The OUT parameter of all the interaction can be represented with the help of five parallel arrays. First array represents the starting vertex of interaction, second array represents the ending vertex of interaction, third array represents the number of standard out parameters passed by the starting vertex, fourth array represents the number of moderate out parameter passed by the starting vertex and fifth array represents number of critical out parameter passed by the starting vertex. The total number of rows of these parallel arrays will be determined by total number of one's exist in the interface matrix.

Srtpoint array[ ]= { v1,v2,v3………………… vn}

Edpoint array[ ]={v1,v2,v3…………………..vn}

Std array[ ]= {1,2,3,4…………………n}

Mod array[ ]={1,2,3,4……………….n}

Crit array[ ]={1,2,3,4………………….n}

ACCOC(Average Component to Component Out Parameters Complexity):

$$ACCOC= \frac{\sum_{i=0}^{m} CCOCi}{m}$$

$$CCOC=\sum_{i=0}^{n}(OSi * Ws) + (OMi * Wm)(OCi * Wc)$$

CCOC= Component to Component OUT Parameters Complexity

OS= standard type of OUT parameter

OM= Moderate type of OUT parameter

OC= Critical type of OUT parameter

Ws, Wm, Wc are weight factors for standard, moderate and critical type of OUT parameters.

Each return value is considered as IN parameter. IN parameter can be of standard, moderate or critical. Interface method either returns a standard type of variable or moderate type of variable or critical type of variable or no value is to be returned by the interface method. The weight factor for standard variable is 0.10, for moderate variable is 0.20 and for critical variable is 0.30

ACCIC(Average Component to Component IN parameters complexity):

$$ACCIC = \sum_{i=0}^{m} \frac{CCICi}{m}$$

$$\sum_{i=0}^{n} \begin{cases} 0.10 \le xi \le 0.30 & if\ IN\ parameter\ exist \\ 0 & if\ IN\ parameter\ does\ not\ exist \end{cases}$$

CCIC= Component to Component IN parameter Complexity

IS= standard type of IN parameter

IM= Moderate type of IN parameter

IC= Critical type of IN parameter

Ws, Wm, Wc are weight factors for standard, moderate and critical type of IN parameters.

ACCC(Average Component to Component Complexity):

ACCC= ACCIC + ACCOC

ACCIC = Average Component to Component IN parameter Complexity

ACCOC = Average Component to Component OUT parameter Complexity

4. CASE STUDY AND EXPERIMENTAL RESULTS

4.1 Cohesion

Suppose there are four components. These components are represented with the help of graph. Graph G(V,E) where V represents vertex and E represents edge.

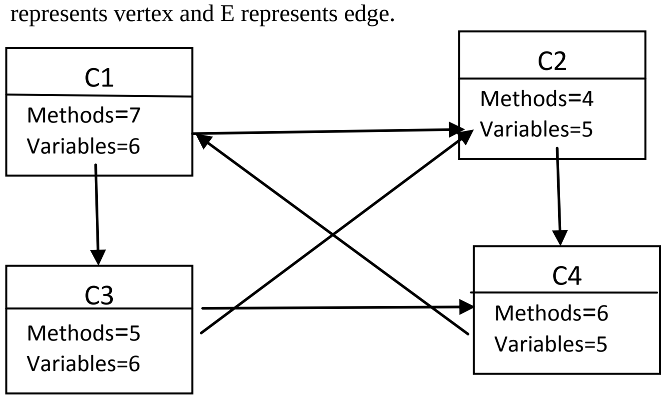


Figure 1 shows components relationship and their methods and variables

The following table shows the component with their method and instance variables.

Here M_c are the methods in a class of a component and V_c are the instance variables of the class. V_{si}, V_{mi} and V_{ci} are the standard variables, moderate variables and critical variables respectively. Fv is the frequency of each variable used by different methods. F_{vsi}, F_{vmi} and F_{vci} are frequency of standard, moderate and critical type of variables. SOM is the sum of methods which are using same type of variables SOM(sv), SOM(Mv) and SOM(cv) are some of methods which are using

.

Table 3 shows the frequency of different type of variables.

Component	Mc	Vc			Fv			SOM		
		V si	V mi	V ci	Fv si	Fv mi	Fv ci	SOM(sv)	SOM(mv)	SOM(cv)
C1	7	6			20			12		
		4	1	1	16	3	1	7	4	1
C2	4	5			5			4		
		1	3	1	1	3	1	1	2	1
C3	5	6			10			6		
		1	1	4	1	1	8	1	1	4
C4	6	5			25			15		
		2	2	1	8	12	5	5	5	5

standard, moderate and critical type of variables.

COVC(c1) = (16*.1 + 3*.2 + 1*.3)/6

= 2.5/6

=0.42

COVC(c2) = (1*.1 + 3*.2 + 1*.3)/5

= 1/5

= 0.2

COVC(c3) = (1*.1 + 1*.2 + 8*.3)/6

= 2.7/6

= 0.45

COVC(c4) = (8*.1 + 12*.2 + 5*.3)/5

= 4.7/5

= 0.94

Here .1, .2 and .3 are the weights of the standard, moderate and critical instance variables.

COMC(c1)= (7*.1 + 4*.2 +1*.3)/7

= 1.8/7

= 0.26

COMC(c2)= (1*.1 + 2*.2 + 1*.3)/4

= .80/4

=.2

COMC(c3) = (1*.1 + 1*.2 + 4*.3)/5

= 1.5/5

= 0.3

COMC(c4) = (5*.1 + 5*.2 + 5*.3)/6

= 3/6

= 0.5

TCCC(c1)= 0.43+0.26

= 0.69

TCCC(c2)= 0.20+0.20

= 0.40

TCCC(c3)= 0.45+0.30

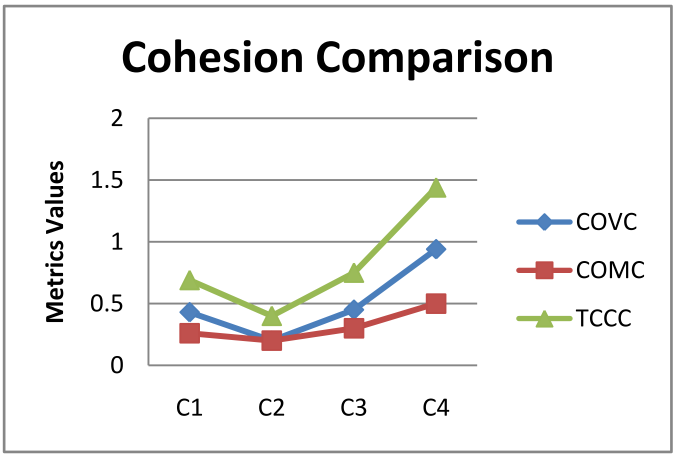
= 0.75

TCCC(c4)= 0.90+0.5

= 1.4

Table 4 shows COVC, COMC and TCCC values

Component	COVC	COMC	TCCC
C1	0.43	0.26	0.69
C2	0.20	0.20	0.40
C3	0.45	0.30	0.75
C4	0.94	0.50	1.44



Graph 1 shows the graphical representation of cohesion metrics

The above line graphs shows values of COVC, COMC and TCCC of different components. X axis represents the components c1, c2, c3 and c4. This graphs shows c2 component which has lowest value of COVC, COMC and TCCC. C4 component contains the highest values for COVC, COMC and TCCC. In c4 frequency of the moderate instance variables are highest as compared to another component. After comparing their result, finding is variation of the result depends on the frequency of instance variables. Cohesion represents the togetherness of variables and methods of the component

4.2 Coupling

Suppose there are four component c1, c2, c3 and c4. Each component have some methods and instance variables. A directed graph G(v,e) represent four components, each vertex(v) in the graph represents the component and each edge(e) represents the interface among the components. Each edge has some OUT and IN parameter which is passed by one component to another component.

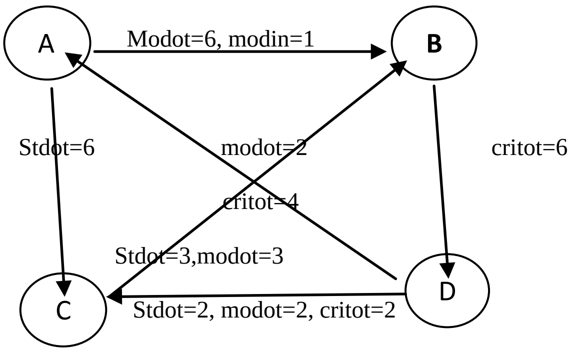


Figure 2 shows relationship between componenets by passing in or out parameters.

An interface matrix represents this graph with the help of interface matrix IM(n*n). In this matrix total num of rows and Columns are equal to the total number of components or vertex. Here IM[I,j] is equal to one if there is an interface between the components and zero if there is no interface among the components.

$$IM[i,j] = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \left| \begin{array}{cccc} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{array} \right| \end{array}$$

We can represent the edge of graph G by parallel arrays

Table 5 shows different parallel arrays showing out and in parameters..

Index	Srtpoint[]	Edpoint[]	Std[]	Mod[]	Crit[]
1	A	B	0	6	0
2	A	C	6	0	0
3	B	D	0	0	6
4	C	B	3	3	0
5	D	A	0	2	4
6	D	C	2	2	2

The first row of the array represents that there is an interface between component A and component B which has six out parameter of moderate type and so on. From these arrays we will find the ACCOC.

ACCOC= $\frac{\sum_{i=0}^{m} CCOCi}{m}$

CCOC= $\sum_{i=0}^{n}(OSi * Ws) + (OMi * Wm) + (OCi * Wc)$

$CCOC_1$= 0*0.10 + 6*0.20 + 0*0.30 = 1.2

$CCOC_2$= 6*0.10 + 0*0.20 + 0*0.30 = 0.6

$CCOC_3$= 0*0.10 + 0*0.20 + 6*0.30 = 1.8

$CCOC_4$ = 3*0.10 + 3*0.20 + 0*0.30 = 0.9

$CCOC_5$ = 0*0.10 + 2*0.20 + 4*0.30 = 1.6

$CCOC_6$ = 2*0.10 + 2*0.20 + 2*0.30 = 1.2

ACCOC= (1.2 + 0.6 + 1.8 + 0.9 + 1.6 +1.2) /4

= 7.3/4

=1.825

ACCIC = $\sum_{i=0}^{m} \frac{CCICi}{m}$

CCIC= $\sum_{i=0}^{n} \begin{cases} 0.10 \le xi \le 0.30 & if\ IN\ parameter\ exist \\ 0 & if\ IN\ parameter\ does\ not\ exist \end{cases}$

$CCIC_1$= 0.20, $CCIC_2$=0.10, $CCIC_3$=0.30, $CCIC_4$=0.30, $CCIC_5$=0.30, $CCIC_6$=0.20

ACCIC= (0.20 + 0.10 + 0.30 + 0.30 + 0.30 + 0.20)/4

= 1.4/4

=0.35

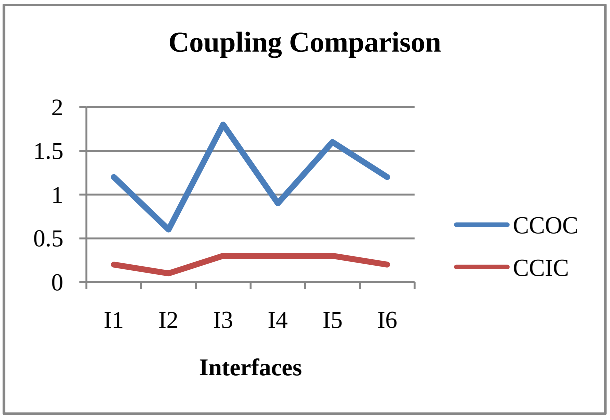
ACCC= ACCIC + ACCOC

ACCC= 0.35 + 1.825

= 2.175

Table 6 shows interface coupling metrics

Interface	CCOC	ACCOC	CCIC	ACCIC	ACCC
I_1	1.2		0.20		
I_2	0.6		0.10		
I_3	1.8		0.30		
I_4	0.9	1.825	0.30	0.35	2.175
I_5	1.6		0.30		
I_6	1.2		0.20		



Graph 2 shows graphical representation of the result of coupling metrics

The above graph shows the different interfaces. Here I2 has the lowest values for CCOC and CCIC and I3 has highest values for CCOC and CCIC. Coupling represents how much one component is dependent on the other component. Coupling should be low. From this I2 has the lowest coupling i.e component A and C are not much dependent on each other.

5. CONCLUSION

In the example there are four components, each component having a class and having some member functions and some instance variables. After comparing their result, finding is variation of the result depends on the frequency of instance variables and type of variables used within component. In the example the value of TCCC of C4 is higher than other components; based on the fact that the frequency of the moderate variables is higher than the standard and critical variables. To make things more clear one can state that if the frequency of the moderate instance variables is more than the frequency of the standard and critical variables within a component then the resulting cohesion value is high which indicates that the complexity of the component is low and hence the reusability factor is better compared to other components.

In standard rule Cohesion should be high and coupling should be low. Cohesion represents the togetherness of variables and methods of the component. Here C4 component represents the higher relatedness of their variables and methods. In coupling there are six interfaces each interface has different number of IN and OUT parameters of different types. In this example the interface2 has lowest value of CCOC and CCIC metrics because interface2 has standard OUT parameters and standard IN parameter. From this the conclusion is to be drawn that the complexity (coupling or cohesion) of the component depends on the frequency of the variables and the type of variables. The result shows that these parameters affect the complexity of the component. The proposed complexity appears to be logical and fits the intuitive understanding but is not the only criteria for deciding the overall complexity of a CBSE. More empirical research by applying our proposed metrics in the real CBSS systems is also one of our future works. Using data from industry implemented projects will provide a basis to examine the relationship between proposed metric values and several quality attributes of CBS.

6. REFERENCES

[1] Umesh Tiwari and Santosh Kumar(2014) "Cyclomatic Complexity Metric for Component Based Software", ACM SIGSOFT Software Engineering Notes page 1 vol 39 No1, Jan 2014

[2] Sonu Mittal and Pradeep Kr Bhatia(2013) "Predicting Quantitative Functional Dependency Metric Based Upon the Interface Complexity Metric In Component Based Software", International Journal of Computer Application, Vol 73, No 2, July 2013

[3] Navneet Kaur, Ashima Singh(2013) "A Complexity Metrics for Black Box Components", International Journal of soft computing and engineering. Vol 3, issue 2,May 2013

[4] Rajender Singh Chhillar and Praveen Kajla(2012) "New Component Composition Metrics for Component Based Software Development", International Journal of Computer Application, Vol 60, No15, Dec 2012

[5] Rajender Singh Chhillar, Priyanka Ahlawat and Usha Kumari (2012) "Measuring Complexity of Component Based System Using Weighted Assignment Technique", 2nd International Conference on information Communication and Management(ICICM 2012).

[6] Usha Kumari and Shuchita Upadhyaya (2011): An Interface Complexity Measure for Component-based Software Systems International Journal of Computer Applications (0975 – 8887) Volume 36– No.1

[7] Jianguo Chen and Hui Wang (2011); Complexity Metrics for Component-based Software Systems; International Journal of Digital Content Technology and its Applications. Volume 5, Number 3

[8] Sengupta, S., Kanjilal, A. (2011): Measuring Complexity of Component Based Architecture : A Graph Based Approach, ACM SIGSOFT Software Engineering Notes, 36 (1), pp. 1-10.

[9] Usha Chhillar, Sucheta Bhasin (2011): A Journey of Software Metrics: Traditional to Aspect-Oriented Paradigm, 5th National Conference on Computing For Nation Development, 10th -11th March, 2011, New Delhi, pp. 289-293.

[10] Chen, Wang, Zhou (2011): Complexity Metrics for Component Based Software Systems, International Journal of Digital Content Technology and its Applications, Volume 5, Number 3, March 2011. Doi:10.4156/jdcta.vol5.issue3.24

[11] Sharma, A., Grover, P.S., Kumar, R. (2009): Dependency Analysis for Component-Based Software Systems, ACM SIGSOFT Software Engineering Notes, 34 (4), pp. 1-6.

[12] V. Lakshmi Narasimhan, P. T. Parthasarathy, and M. Das (2009): Evaluation of a Suite of Metrics for Component Based Software Engineering (CBSE), Issues in Informing Science and Information Technology Volume 6, 2009

[13] Gill, N.S, Balkishan (2008): Dependency and Interaction Oriented Complexity Metrics of Component-Based Systems, ACM SIGSOFT Software Engineering Notes, 33 (2), pp. 1-5.

[14] Gui, Scott,(2008) New Coupling and Cohesion Metrics for Evaluation of Software Component Reusability, 9th International Conference For Young Computer Scientists, IEEE 2008. DOI 10.1109/ICYCS.2008.270

[15] Li, X, Liu, Z. Pan, B. and Xing, B.(2001) A Measurement Tool for Object Oriented Software and Measurement Experiments with IT. In Proc. IWSM 2000. (Lecture Notes in Computer Science 2006, Springer- Verlag, Berlin, Heidelberg, 2001),44-54

[16] W. Kozaczynski, G. Booch (1998), "Component-Based Software Engineering," IEEE Software Volume: 155, Sept.-Oct. 1998, pp. 34–36.

[17] Biemen, J. M. and Kang, B-Y. Cohesion and Reuse in an Object-Oriented System. In Proc. ACM Symposium on Software Reusability (SSR'95). (April 1995) 259-262

[18] Hitz, M. and Montazeri, B. Measuring coupling and cohesion in object oriented systems. Proceedings of International Symposium on Applied Corporate Computing. (Monterrey,Mexico, 1995).

[19] Chidamber, S.R. and Kemerer, C.K. towards a Metrics Suite for Object Oriented Design. Proceedings of 6th ACM Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA'91), (Phoenix, Arizona,1991), 197-211.