International Journal on Emerging Technologies 10(1): 211-217(2019)

ISSN No. (Print) : 0975-8364
ISSN No. (Online) : 2249-3255

Optimization of Software Package Selection using Cohesion Measurement and Complexity Metric for CBSS Development

Iyyappan M.[1] and Arvind Kumar[2]
[1]*Department of Computer Science & Engineering, SRM University, Sonepat (Haryana), India.*
[2]*Department of Computer Science & Engineering, SRM University, Sonepat (Haryana), India.*

(Corresponding author: Iyyappan M.)
(Received 10 April 2019, Revised 06 June 2019, Accepted 12 June 2019)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Component based software engineering play a vital role for the development of software because its supports higher level of program code and system level of maintenance. CBSE is concerned about the building of software which is able to satisfy a client-specific requirements with the use of reuse or independent development. For metric measurement of software using the various factors like a complexity, quality and reliability. Coupling and cohesion measurement are mainly used for analysing the better quality of software design, increased reliability of high cohesion & low coupling and reduced the complexity of the system software. Complexity metrics are mainly concern about the calculating the relationship between software packages, group of classes and Sub classes of methods. To show the improving aspects of software system like a reduced complexity and increased higher reliability. The software packages of reusable components are related together of classes and methods, for supporting the enhanced technology object oriented programing languages of software systems. To promote reuse of Component based software engineering in object oriented system mainly focus on the development, implementation and maintenance phases. To make a phases for easy understanding by the user with cohesive measurement of packages, classes and methods. Quantification value & analysis method of cohesion metric, for the software packages can be very useful for assessing their component reusability, quality, complexity and reliability etc. In this modules, a new set of metric calculation are approached for the complexity of low cohesion and complexity of high cohesion of the software package, classes and methods are proposed. High cohesion and low cohesion measurement using degree of inter-dependence among the package of module. The hierarchical structure of the package relation among the elements, classes and methods are used for measurement. The proposed cohesion metric of real data set values has been analysed theoretical as well as experimental result also observed the optimized result for better performance of software system. An empirical study has been conducted using 25 various software packages, it has been taken from the six different free-source code of java programming used for the software projects developments. The proposed package of the, complexity of low and complexity of high cohesion measurement is found to be improved result of the quality factor as well as optimized line of code for reusability packages. This result comparatively better than the existing cohesion measurement of reusable in component based software engineering.

Keywords: Cohesion component, Metrics measurement, Improved quality, Better reusability, Software packages, System complexity – Low, High, Object oriented programming languages, Component Based Software Engineering.

I. INTRODUCTION

In the software development for the system maintenance mainly focus on the cost. Cost depends on the quality of the system and complexity of the system [1]. Analysing and Design efforts [2] are not useful on the phases of maintenance because it has a more complexity and less reliability of the system. So before moving to the implementation process better analyse the complexity and reliability of the system. In hierarchy process of object oriented system for the packages mainly focus on the Classes and methods. For the metrics measurement of the cohesion are used packages and its various elements of the classes or interfaces [3]. Cohesive packages provide a complete structure for the classes to make a proper functionality which helps to maintain the system and increase reusable of the system. In object oriented programming languages mainly focus on the classes, attribute and operation all are related to the packages. The packages will provide the major entities is a type of concrete relationships of class entities [4, 5]. The paper mainly focus on the metrics measurement and reliability measure for the implementation of component based software engineering environment [6]. Existing proposed metric are used and validated on a real data set for improving the result of the package cohesion metrics measurement. “You cannot control what you cannot measure” [7], it’s mentioning about the Component measurements. The quality and maintenance of the software system will go for the new development of component or existing developed component for the system adaptation. The packages are main concern about the classes and package cohesion. The four major components are there In-house components, Commercial off-the shelf Component, Open source software component and off-the shelf component. Existing methodology only focus on the In-house and OSS concept because we can use source code with-out modification[8].Only a few studies have been conducted on the usage of reliability metric on a real data set. In this paper, a new cohesion metrics to provide a better quality of the software packages. It

also verified and validated with the existing approach of the package level of component. The improved better results are observed from new proposed metrics. In this paper major concern about the basics of software component and properties of object oriented systems also define the cohesion measure which is taken at package level of Low & High cohesion for the object-oriented systems. The [9] Briand.et.al existing theoretical approach used for our proposed the analysis and measurement of metrics low cohesion as well as high cohesion components. In Briand methodology used correlating coefficient factor to measure the external quality factor for reusability of component based software engineering. This proposed paper content contain following topics, In Section. 2 about the literature survey of existing work & ideas and section 3 provide Cohesion Package Measurement on Theoretical Approach which is taken for proposing the metric CLC and CHC [10]. In Section 4 contain Cohesion package measurement of complexity on theoretical validation of proposed measure. Section 5 Complexity of Cohesion Packages for the Metric Proposed Experimental Validation using the software project of public available source code are used for the comparison of CLC & CHC metric. In section 6 Proposed Analysis methodology used to measure Coefficient Correlation. Section 7 presents the conclusions.

II. RELATED WORK

In the review of literature survey, various metrics measurement of cohesion methods are developed by considering the modules as well the classes [11] of the design level focused on the cohesion metrics. In this paper used various functionalities of the cohesion metrics for the reusability of component based software engineering [12].In this paper analysing the quality of software package level on the basis of object oriented design. In the software design must have higher cohesion and more reusability. This design framework assessing the class level of attribute for the package cohesion of aspect oriented system [5]. In this paper, we perform an empirical analysis on Python packages for the two measures namely coupling and cohesion. The Structural methods of classes like a complexity, line of code, coupling and cohesion these factors are measured from the system maintenance and software reusability[8]. In real data set value used from the Jakarta apache organization [14] using Perl programming languages for the website development. In this empirical study discuss about the sensible analysed cohesion metric and deeper investigate search of the system [15]. Coupling and Cohesion metrics are used for measuring procedure and object oriented programming languages. [16] Coupling describes the interdependency between classes and methods. Cohesion describes about the binding of relationship among various elements and methods. That component bindings are related within module of the object class. The decomposition of a large program into modules can be guided by the use of a property called cohesion. In this paper proposed as a discriminant for classifying modules to set the metric value ranges for module classification for the cohesion [17]. The Comprehensive framework takes in to account of distinction between the object level and class level coupling [19] in this methodology which is depends on the dynamic dependencies and static dependencies for the maintenance phase[20].In this paper represent an experimental study to validate the modified global metrics by showing their relationship to maintainability and testability. The prediction model for quality attribute measurement of new set of metrics to improve the result [21].A measure proposed by Emerson to compute cohesion by considering of Pascal procedures [18].This proposed measure was based on the graphic theory for computing the relationships between the elements of the class. The chidamber proposed to measure weighted method per class with total number of methods and weighting scheme. The prioritization of methods is used for the module cohesion measurement that is based on the slice of program code was proposed by Bieman and Ott [13]. Another method also proposed by Bieman and Kang, Cohesion of class has been tightly bounded with each other and loosely bounded cohesion for the class, are used for the metric measurement. A many number of methods or measures proposed for In-house or OSS components are summarized below:-the various research gap in the [27] Chidamber & Kemere proposal, weighted method per class mainly focus on the total number of methods and priority of the method. To find the average calculation of metric in object oriented design. Implementation phases are lacking on this methodology. Also coupling measurement of kemerer found some research gap like ametric used to measure the coupling and cohesion. It is used analyse the overall classes are present in the coupling measurement. Major focus on the loose Coupling and Tight Cohesion. Hassoun proposed work, The Major concern for this dynamic coupling metric how much duration spend on the time limit for the particular measurement of coupling between classes. J. Chen [28] research gap like a Cohesion measurement for the software system on the basis of complexity. But weyuker methodology are not able to utilize on this paper. S. Patel and J. Kaur [29] found a research gap, this paper will provide a basic information about their component and internal structure of methods or attributes. Coupling between components are using the sharing of components.

III. COHESION PACKAGE MEASUREMENT ON THEORETICAL APPROACH

In the following modules we used the Inheritance of hierarchy steps like a tree structure which is related to the packages, classes and methods similar to the object oriented programming.

A. Definition of Software packages, Group of classes and methods

On this cohesion measurement main concern about the encapsulated about the group of various classes related to the packages, sub-packages of among the variable and interface relationship between methods. Generally a software is made up of thousand number of the source code and for placing the appropriate interfaces and methods in to appropriate way packages are required.

B. Empty Packages

When a package is having no element so no relations exist between the elements therefore termed as an empty package and in that case the cohesion value can be considered as zero. Classes can be called as bunch of objects or it's a way with the help of which objects can be defined.

C. Complexity of Low cohesion (CLC) and Complexity of High cohesion (CHC)

Software engineering domain focus on the two major types of the development: Component based software

engineering (CBSE) and Commercial-off the shelf (COTS) development. In the component based software engineering are mainly concern about complexity metric and reliability metrics. For this measurements are observed the quantitative result from development and maintenance of the software. The more number of the research focused on complexity measurement between the coupling and cohesion metric values.

For measuring the complexity of java programming packages used the group of classes, sub packages of variable and interface among the methods. For selecting the low coupling and high cohesion result is not enough to choose suitable component from the complexity. In this proposed paper used the concepts, optimal selection of complexity component between the coupling and cohesion [10]. This selection component provide an improved performance, higher quality and increased reliability. The following steps are applied for the calculation of complexity among the component and various packages.

1) Considering the real data set for cohesion measurement.
2) Finding the parameters like R[D], R[DUI], and CDI.
3) Calculating the package measurement between the Complexity Low cohesion and Complexity High cohesion.

These result comes as a comparison between low cohesion and high cohesion. The improved result of CLC and CHC will provide the better quality of software as well as less complication factor among the software development and maintenance. Also we followed the terminology of Low coupling and High cohesion between the components based software engineering. Parameters used in the metric are defined as follows:-

R[D]:-The Client direct request for low cohesion measurement between group of classes and subclasses among the methods.

R[DUI]: - In this Client request of direct and indirect for high cohesion measurement on the basis of complexity.

CDI: - The overall direct or indirect connections are measured from this case study like an $\mathbf{P^a\ (E^{a+1},\ r^{a+1})}$ / **R(R-1)** to measure a binary directed relationship between classes of elements '**E**' and Classes of relation '**r**' at the hierarchy level a+1 on the packages. Then R is representing as a client request of total number of packages in the case study. In the package measurement of hierarchical level represent the $\mathbf{E^{a+1} = 1}$ and $\mathbf{r^{a+1} = 1}$, so we are able to measure the client direct request of direct or indirect connections**2** / **R(R-1)** has been taken from the concept of TCC (Tight class cohesion) and LCC (Loose Class Cohesion) metric.

CLC & CHC: - Package measurement on the basis of Complexity of Low cohesion component & Complexity of High cohesion component.

$$LC\ \&\ CHC = \begin{Bmatrix} 0 & if\ (n = 0) \\ \frac{R(D)}{CDI} & if\ (n < 1) \\ \frac{R(DUI)}{CDI} & if (n > 1) \\ 1 & if\ (n = 1) \end{Bmatrix}$$

Where 'n' represent the number of elements between the classes and methods. On other cases:-If n=0, it represent the no element value over there, so there is no possibility of the relation therefore computed value of CLC & CHC is also 0.If n=1 means that the single element value is existing over here, so the relation existing will also be single, hence the value of CLC & CHC is also 1.

Table 1: Illustration of package measurement for Complexity of Low cohesion & Complexity of High cohesion versus Package cohesion metric(PCOh) [22].

package myshapes;
public interface Drawable
{ public void
draw(Graphics g);
}
class Line implements
Drawable { public void
draw(Graphics g) {
. . . // do something –
presumably, draw a line
. . . // other methods and
variables
}
	package
	myshapes.round;
	import
	myshapes.Drawable;
	public class Circle {
	public void findArea( ) {
	. . . // find area of circle
	.. . // other methods and
	variables
	}
	Class FilledCircle extends
	Circle
	implements Drawable{
	public void draw(Graphics
	g) {
	. . . // do something –
	draw a filled Circle
	}
	void findArea( ) {
	. . . // find area of filled
	circle
	}
	. . . // other methods and
	variables
	}

IV. COHESION PACKAGE MEASUREMENT OF COMPLEXITY ON THEORETICAL VALIDATION

In this section mainly focus about the Complexity measurement of cohesion packages for the component also use to measure the metric value of low complexity as well as high complexity which is derived from Briand four properties [22]. Weyuker's also proposed a framework for measuring a complexity metric to get accurate the accurate result from the observation of validate real data set value. Some other evaluation frameworks such as Zuse framework [23] and Tian & Zelkowitz [24] axioms are also used for validation of complexity measures. But Briand *et al*. methodology mainly focus on the four properties which is used for measuring the complexity metric value.

Property 1:Set of non-negative integer value and rating of normalization scale. - According to the above explanation given in the summarization of proposed measure the computed value of CLC and CHC belongs to a specified interval of [0, 10] & [0, 20]. Therefore the value computed by using above already defined that the measure will be always with zero & positive number as well as regular number.

Property 2: Empty assessment number and highest assessment number - Whenever the component is empty then the value assigned to be as 0 because no relations between the packages classes and methods are existing in such a case so null value property is satisfied. If the component is having a single element then some types of relations between the packages, classes and methods are existing and the maximum value defined to be 1.Hence the measurement satisfies on the property 2.

Property 3: Isotone subset of the real numbers

According to Briand et al. framework, this property declares that adding relations will not decrease cohesion. Since the adding of relations in to the component will increases the value of R [D] but

simultaneously the value of DUI get also increased as (DUI=direct connection + indirect connection).Hence adding more number of relations also increase the cohesion value and hence it satisfies the property of monotonicity.

Property 4: Combine the not connected packages - According to this property the joining of two unconnected packages should not increase the value of cohesion. The software package cohesion component complexity metric satisfies this property by the fact that if the packages are added or merged the value in the denominator will also increase with the value in the numerator and also the calculation of parameters are totally based on number of packages not on the merging or the connection of packages.

V. COMPLEXITY OF COHESION PACKAGES FOR THE METRIC PROPOSED EXPERIMENTAL VALIDATION

In this section we are focusing on the experimental validation of Complexity metric measurement for Low cohesion and High cohesion of packages and showing the improved results of reliability and complexity on the impact of component reusability of system software. This method is used for improving the quality and reducing the complexity of our cohesion packages.

A. Experiment analysis for proposed measurement

The empirical study of the PCoh metrics are used for analysing the complexity and reliability of the system. In our proposed methodology also appeared those metric measurement for improving the better performance of the system. The template or guidelines provided for defining the measurement goal [25]. This goal consist of: **Goal of this preliminary study**: Number of packages and the relation between the packages, classes and methods. **Motive of this paper**: analysis and comparison with the PC ohmetric values. **Improved Factors**: For using the reusable component the system quality also increased for software developments. **Observed empirical study:** Software developer have observed that the selection of suitable components. **Software Elements:** free available source code of Java projects are available on the open-source. These above defined goals help us to determine the type of real data to be collected.

B. Empirical hypotheses for significant correlation

On this experimental topics we observed that the hypothesis raise the question between the true or false. If it's not a true, approaching the similar statistical result equivalent to the true conditions. To get the result of more than one attribute from this objective study to improve the quality factor[26]. In this case, the statistical result observed between the packages, group of classes and Subclasses of methods for the component reusability. The statistical analysis is based on the various hypothesis:***Hyp*0: *q*=0 (Empty value)** - There is no significant correlation between the low cohesion and high cohesion among the componentpackages.***Hyp*1: *q*≠0 (Equivalent value)** - It is significant correlation between the proposed high cohesion and low cohesion of complexity measurement for reusability.

C. Empirical environment for the freeware code of software projects

There are a various number of free source code software projects are available on the online website for a many number of purposes. "The sample used for this experimental study was taken from six open-source software projects whose source code was readily available for use. The major reason behind selection of these projects was that these software projects were developed in Java and were organized using packages. The presence of packages in these projects made it possible to apply package level metrics on them. Twenty-five pack-ages taken from six open source projects were used in order to experimentally evaluate the proposed metric. Out of these six projects, four belonged to the Apache soft-ware foundation and eighteen packages belonging to these four projects were downloaded from Apache Jakarta web-site" [22,27].

VI. PROPOSED ANALYSIS METHODOLOGY OF COEFFICIENT CORRELATION IN COHESION PACKAGES

A mathematical analytics are proposed to correlate with the system software packages of Low cohesion component and High cohesion component metric for reusability of component based software and also the computed values are also compared with the Pcoh metric for examine the improved results. For the empirical validation of metrics correlation is the suitable technique. Li and Henry studied the various metrics and also correlate them to determine their effectiveness [28]. Karl person Correlation coefficient are used for measuring correlation among the probability value. This method assumes that there is a constant linear relationship between the two different variables. In this relationship one of the variable represent as an independent module and other is a dependent relationship module. Karl Pearson's coefficient of correlation is given by [29]:

$$r = \frac{\Sigma(Xi - \bar{X})(Yi - \bar{Y})}{n * \sigma x * \sigma y}$$

In this correlation measurement has taken place of variable like an i^{th} term and mean calculation with the help of median measurement among the coefficient factor. Also derived the standard deviation among the 'n' number of component for calculating the correlation coefficient. This correlation consider of -1 as a negative value relation, +1 consider as a positive relationship variable and 0 it's like a no linear variable. The proposed measure of low correlation and high correlation are determined by the various coefficient factors [22].

The ratings for the cohesion can be defined as:-

Table 2: Measure Coefficient Correlation.

Complexity of Low cohesion	Complexity of High Cohesion
0 "minimum"	0 to 1.0 "minimum"
1.0 to 2.0 "just above the minimum"	2.0 to 3.0 "just above the minimum"
2.0 to 4.0 "average"	3.0 to 6.0 "average"
4.0 to 5.0 "Best"	7.0 to 9.0 "Best"
5.0 to 8.0 "optimum"	10 to 18.0 "optimum"

Table 3: Line of code measurement on the basis of Packages and classes.

S. No.	Name of the Software Projects
1.	Byte Code Engineering Library (BCEL)
2.	Bean Scripting Framework (BSF)
3.	Jakarta-ORO
4.	Element Construction Set (ECS)
5.	XGen Source Code Generator
6	Java unit (JUnit)

We are using the following facts as:-
Number of elements consisting of interfaces and subpackages.so Total number of classes-no of elements = filtered classes. Now because in this real data set only a single package is used in a single component so the direct connection between the classes and methods are the filtered classes.

Table 4: Calculate Values of Complexity of Low Cohesion measure.

S. No.	Name of the Software Packages	TC	NE	D	NDIUC	CLC
1.	org.apache.bcel.verifier	48	14	34	56	0.607
2.	org.apache.bcel.verifier.exc	14	14	0	10	0
3.	org.apache.bcel.verifier.statics	9	9	0	1	0
4.	org.apache.bcel.verifier.structurals	14	14	0	10	0
5.	org.apache.bcel.util	20	20	0	10	0
6.	org.apache.bsf.util.event	21	6	15	18	0.833
7.	org.apache.bsf.util.event.generator	4	4	0	4	0
8.	org.apache.bsf.util	54	20	34	30	1.13
9.	org.apache.bsf.util.type	2	2	0	1	0
10.	org.apache.bsf.util.cf	2	2	0	1	0
11.	org.apache.oro.io	4	4	0	3	0
12.	org.apache.oro.text	50	15	35	34	1.02
13.	org.apache.oro.text.awk	17	17	0	41	0
14.	org.apache.oro.text.perl	3	3	0	0	0
15.	org.apache.oro.util	7	7	0	11	0
16.	org.apache.ecs.jsp	15	15	0	14	0
17.	org.apache.ecs.storage	3	3	0	3	0
18.	org.apache.ecs.xml	3	3	0	2	0
19.	workzen.xgen.ant.legacy	4	4	0	3	0
20.	workzen.xgen.engine	2	2	0	1	0
21.	workzen.xgen.loader	7	7	0	7	0
22.	junit.samples	2	4	2	1	2
23.	junit.samples.money	4	4	0	9	0
24.	junit.tests	36	5	31	4	7.75
25.	junit.tests.extensions	5	5	0	4	0

Table 5: Calculate Values of Complexity of High Cohesion measure.

S.No.	Name of the Software Packages	TC	NE	D	NDIUC	CHC
1.	org.apache.bcel.verifier	48	14	62	56	1.107
2.	org.apache.bcel.verifier.exc	14	14	28	10	2.8
3.	org.apache.bcel.verifier.statics	9	9	18	1	18
4.	org.apache.bcel.verifier.structurals	14	14	28	10	2.8
5.	org.apache.bcel.util	20	20	40	10	4
6.	org.apache.bsf.util.event	21	6	27	18	1.5
7.	org.apache.bsf.util.event.generator	4	4	8	4	2
8.	org.apache.bsf.util	54	20	74	30	2.46
9.	org.apache.bsf.util.type	2	2	4	1	4
10.	org.apache.bsf.util.cf	2	2	4	1	4
11	org.apache.oro.io	4	4	8	3	2.66
12.	org.apache.oro.text	50	15	65	34	1.911
13.	org.apache.oro.text.awk	17	17	34	41	0.829
14	org.apache.oro.text.perl	3	3	6	0	0
15	org.apache.oro.util	7	7	14	11	1.27
16.	org.apache.ecs.jsp	15	15	30	14	2.14
17.	org.apache.ecs.storage	3	3	6	3	2
18.	org.apache.ecs.xml	3	3	6	2	3
19.	workzen.xgen.ant.legacy	4	4	8	3	2.66
20.	workzen.xgen.engine	2	2	4	1	4
21.	workzen.xgen.loader	7	7	14	7	2
22.	junit.samples	2	4	6	1	6
23.	junit.samples.money	4	4	8	9	0.88
24.	junit.tests	36	5	41	4	10.25
25.	junit.tests.extensions	5	5	10	4	2.5

A. Analysis of experimental results of Proposed Low complexity and high complexity

Table 6 shows the package name with the total number of classes, number of elements with their calculated filtered class and it is termed as R(D) & R(DUI).The number of relations is termed as CDI and finally with all these the Complexity of Low and High cohesion is calculated with the help of Package.

Table 6: Complexity of statistical parameter for Low cohesion and High cohesion metric.

Statistical parameter	CLC	CHC
Highest assessment	7.75	18
Lowest assessment	0	0
Median	0	2.5
Mean	0.453	3.39
Standard Deviation	1.53	3.572

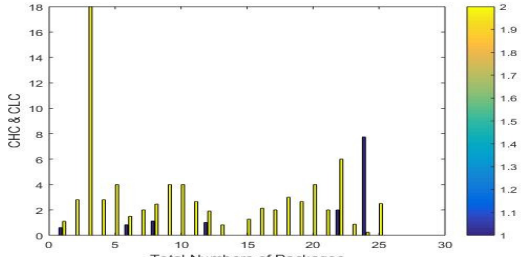
B. Ratings for the component reusability with analysed measure

Below the values in the table are showing that the metric Package cohesion component complexity metric is efficient.

Table 7: Rating of Component with Package.

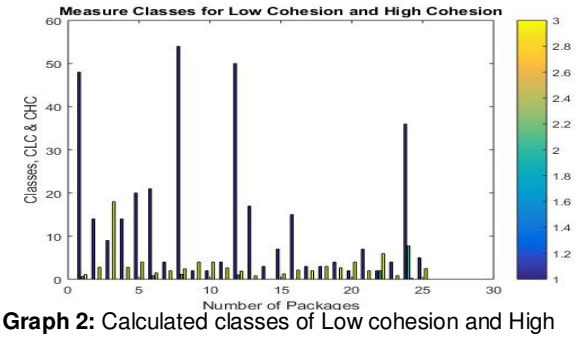
S.No	Name of the Software Packages	Ranking of complexity component
1.	org.apache.bcel.verifier	8
2.	org.apache.bcel.verifier.exc	4
3.	org.apache.bcel.verifier.statics	4
4.	org.apache.bcel.verifier.structurals	3
5.	org.apache.bcel.util	2
6.	org.apache.bsf.util.event	11
7.	org.apache.bsf.util.event.generator	8
8.	org.apache.bsf.util	1
9.	org.apache.bsf.util.type	18
10.	org.apache.bsf.util.cf	10
11.	org.apache.oro.io	13
12.	org.apache.oro.text	3
13.	org.apache.oro.text.awk	1
14.	org.apache.oro.text.perl	2
15.	org.apache.oro.util	1
16.	org.apache.ecs.jsp	5
17.	org.apache.ecs.storage	9
18.	org.apache.ecs.xml	8
19.	workzen.xgen.ant.legacy	5
20.	workzen.xgen.engine	16
21.	workzen.xgen.loader	6
22.	junit.samples	2
23.	junit.samples.money	11
24.	junit.tests	7
25.	junit.tests.extensions	6

From above results it can be concluded that the components having the high values of cohesion associated with their packages are termed as optimum components.

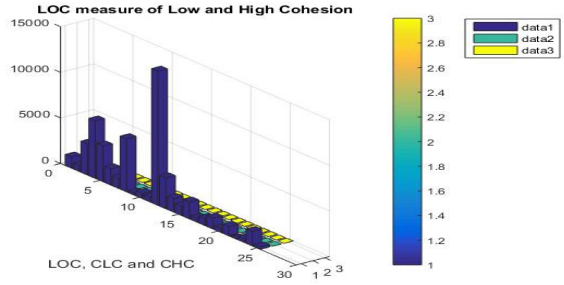


Graph 1: Calculated Complexity of Low cohesion and High cohesion with total number of packages.

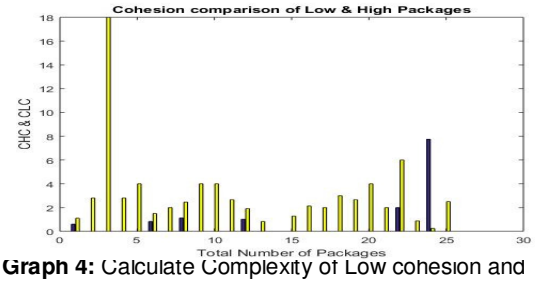
From the table 8, the observed result value between the correlation coefficient and significant values show the better results than the comparing of existing metric measurements.



Graph 2: Calculated classes of Low cohesion and High cohesion with total number of packages.



Graph 3: Calculated Complexity of Low cohesion and High cohesion with Line of code along with number of packages.



Graph 4: Calculate Complexity of Low cohesion and Complexity of High cohesion comparison between total numbers of packages.

Table 8: Comparison with the Correlation coefficient values of Low and High other metrics.

Parameters	CLC	CHC	LCOM	LCOM1	ICH	SCC
Correlation Coefficient	0.20	0.48	-0.32	-0.34	0.12	0.27
Significance value	0.05	0.05	0.05	0.05	0.05	0.05

It is opted for choosing the hypothesis value of 1, not the value of 0, comparison among the less than or greater than value should be always 1, not an empty value hypothesis. So it has been analysed the improved relation between the low & high cohesion complexity metric and component reusability.

VII. CONCLUSIONS AND FUTURE SCOPE

In this proposed concept, it has been analysed between the Low package and High Package cohesion complexity metric, as well as metric outcome also compared with the previous Package cohesion. The CLC and CHC is also validated using those proposed properties in this paper. The proposed measurement

ratio on the basis of direct and indirect relations among the group of classes and subclasses. The standard hierarchal structures of package have also been taken in to consideration. We believe that this metric will help the other developers and OSS users for the calculation of complexity based on the concept of cohesion. In future scope rather than reusability, this metrics are proposed to establish the relationship between the component and package modules, along with these factors also included like a maintainability and adaptability are utilized for the component based development environment. Also coupling & cohesion metric values are reduced the complexity and increase the performance of the Component based software system.

REFERENCES

[1]. Basili, V.R. (1997). Evolving and packaging reading technologies. *J System Software*, **38**(1): 3–12.

[2]. Corbi, T.A. (1989). Program understanding: challenge for the 1990's. *IBM System J.,* **28**(2): 294–306.

[3]. Patel, S., Chu, W.C., and Baxter, R. (1992). A measure for composite module cohesion. In: *Proceedings of the 14th International Conference on Software Engineering,* 38–48.

[4]. Martin, R. (2002). Agile software development, principles, patterns, and practices. Prentice-Hall, New York.

[5]. Kaur, P.J., Kaushal, S., and Sangaiah, A.K. (2018). A Framework for Assessing Reusability Using Package Cohesion Measure in Aspect Oriented Systems. *International Journal of Parallel Programming,* **46**(3): 543–564.

[6]. Telles, M. (2001). C# Black Book. The Coriolis Group.

[7]. DeMarco, T. (1982). Controlling software projects. Yourdon Press, New York.

[8]. Priyalakshmi, G., and Latha, R. (2018). Evaluation of Software Reusability Based on Coupling and Cohesion. *International Journal of Software Engineering and Knowledge Engineering,* **28**(10): 1455-1485.

[9]. Briand, L., Morasca, S., and Basili, V. (1996). Property-based software engineering measurement. *IEEE Trans Software Engineering,* **22**(1): 68–86.

[11]. Bieman, J.M., and Kang, B.K. (1995). Cohesion and reuse in an object-oriented system. In: Proceedings of the symposium on software reusability (SSR'95), Seattle, WA, 259–262.

[12]. Bieman, J.M., and Kang, B.K. (1998). Measuring design-level cohesion. *IEEE Trans Software Engineering,* **24**(2): 111–124.

[13]. Bieman, J.M., and Ott, L.M. (1994). Measuring functional cohesion. *IEEE Trans Software Engineering,* **20**(8): 644–658.

[14]. Briand, L., Morasca, S., and Basili, V. (1996). Property-based software engineering measurement. *IEEE Trans Software Engineering*, **22**(1): 68–86.

[15]. Byte Code Engineering Library (BCEL). (2011). http://jakarta. apache.org/bcel/index.html.

[16]. Counsell, S., Mendes, E., and Swift, S. (2002). Comprehension of object-oriented software cohesion: the empirical quagmire. *In: Proceedings of the 10th international workshop on program comprehension*, 33–42.

[17]. Eder, J., Kappel, G., and Schre, M. (1992). Coupling and cohesion in object-oriented systems. In: Proceedings of the conference on information and knowledge, ACM Press, New York.

[18]. Emerson, T. (1984). A discriminant metric for module cohesion. *In: Proceedings of the 7th international conference on software engineering (ICSE).*

[19]. Chhillar, RS. and Kaijla, P. (2011). A New Knot Model for Component Based Software Development. *IJCSI*, **8**(3): 480.

[20]. Hitz, M., and Montazeri, B. (1995). Measuring coupling and cohesion in object-oriented systems. In: Proceedings of the third international symposium on applied corporate computing (ISACC'95), Monter-rey, Mexico, 25–27.

[21]. Ott, L., Bieman, J.M., Kang, B., and Mehra, B. (1995). Developing measures of class cohesion for object-oriented software. *In: Proceedings of the annual Oregon workshop on software metrics (AOWSM'95).*

[22]. Almugrin, S., Albattah, W., and Melton, A. (2016). Using indirect coupling metrics to predict package maintainability and testability. *Journal of system and software,* **121**: 298-310.

[23]. Zuse, H. (1991). Software complexity: Measures and Methods. Journal Software Testing, Verification and Reliability, **1**(3): 41-42.

[24]. Solingen, R.V. (2002).The goal/question/metric approach, encyclopaedia of software engineering-**2**: 578–583.

[25]. Briand, L., Morasca, S., and Basili, V. (2002). An operational process for goal-driven definition of measures. *IEEE Trans Software Engineering,* **12**: 1106–1125.

[26]. The Apache Jakarta Project. (2011). http://jakarta.apache.org.

[27]. Kothari, C.R. (2007). Research methodology: methods & techniques, revised second edn. New Age International, New Delhi, 139–141.

[28]. Chidamber, S.R., and Kemerer, C.F. (1994). A Metrics suite for object oriented Design. *IEEE Transactions on Software Engineering*, **20**(6): 476-493.

[29]. Chen, J., Wang, H., Zhou, Y., & Bruda, S. D. (2011). Complexity metrics for component-based software systems. *International Journal of Digital Content Technology and its Applications*, ***5***(3): 235-244.

[30]. Patel, S., and Kaur, J. (2016). A Study of component based software system metrics. *International Conference on Computing, Communication and Automation (ICCCA)*, 824-828.

How to cite this article: Iyyappan, M. and Kumar, A. (2019). Optimization of Software Package Selection using Cohesion Measurement and Complexity Metric for CBSS Development. *International Journal on Emerging Technologies*, **10**(1): 211-217.