Package Based Cohesion Measurement in Component Based Software Development

Dr Puneet Goswami[1] and Ms Jyoti Sharma[2]
Professor (CSE), SRM University, Delhi-NCR, Sonipat, Haryana, India
goswamipuneet@gmail.com
Ph.D (CSE) Research Scholar, SRM University, Delhi-NCR, Sonipat, Haryana, India
Kaushik.jyoti27@gmail.com

Abstract The good software design is based on low coupling and high cohesion. Component based software engineering is based on the concept of component reuse but Every time it is not possible to adapt the component according to user requirementsso it's better to check the reliability of components before using them. Packages are re-usable components for mostof object-oriented systems. To promote reuse in object-oriented systems and to make deployment and maintenance tasks easy, the standard relationship between the packages, classes and methods must be cohesive. Identification of reliable component can help to find the reusable components and also affecting the quality of software in component based development environment. In this pa-per, a new measure for the measurement of package cohesion is used on a real data set for the better results. The cohesion of a component is measured by considering the standard relationship between the packages, classes and methods. The hierarchical structure of packages has also been taken into account during the measurement. The used measure has been validated theoretically as well as empirically. An empirical study has been conducted using 25 packages taken from six open-source software projects developed in Java.

Keywords Cohesion**,**Metrics**,** Quality**,** Reusability·Packages, CBSE

Introduction

The cost of a software system is mainly depending on the complexity, maintenance and quality. During maintenance, a lot of analysing and reading efforts are wasted due to high complexity and less reliability of system [1, 2].It would be better to check the reliability and complexity of the components before integrating in to the system. The standard hierarchy consist of packages followed be classes and then methods. Packages consist of many numbers of elements which can helps us for determining the cohesion [3]. Packages are imported and considered to be an important one because they are concrete entities and also the main entities in object oriented programming languages [4, 5, and 6]

This paper primarily targets measuring or to make a estimation of the reliability of component which has to be integrated in to the component based software environment. For this reason the already proposed metric has been used and validated on a real data set for improving the results. "You cannot control what you cannot measure" [7], stresses the need of such measures.The component based software system is basically having the four types of components as In-house, COTS, OSS, OTS (COTS+OSS) and the already proposed metric focuses on In-house and the OSS components where

The source code is available but without modification it can be used [8]. Only a few studies have been conducted on the usage of reliability metric on areal data set. In this paper, a new metric which is already proposed for measuring cohesion at the package-level in order to achieve good quality packages is validated and compared with the already defined package cohesion metric for improving or giving the better results.

This paper first provides the basics and also the properties of object oriented systems and then defines cohesion measure which is taken at package level for object-oriented systems. While defining the metric for cohesion at package level, hierarchical structure of the pack-ages has also been taken into consideration. The already proposed measure is validated theoretically using Briand et al.'s evaluation criteria [9] and usefulness of the proposed metric is also established by correlating this metric with the external quality factor-reusability. This paper is organized as follows. In Sect. 2, the previous related works are reviewed and Sect. 3providethe summarization of approach which is taken for proposing the metic PC_3M [10]. Section 4 presents the theoretical validation of already the proposed measure. Section 5validates the pro-posed measure through experimental evaluation conducted using open-source software projects and also the comparison with the PCOH metric and Section 6 presents the conclusions and future work directions.

Related work

In literature, various cohesion metrics have been developed by considering the modules as well the classes[11,12,13,14,15,16,17,18,19,20].A measureproposed byEmerson 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. Weighted method per class metric proposed by chidamber based on the counting of number of methods in the class and weighting scheme was used at that time for the prioritization of methods that which have to usefirst

The approach of mod-ule cohesion measurement that is based on data slices was proposed by Bieman and Ott [13]. Bieman and Kang came up with another metric defined as TCC(Tight class cohesion and LCC(Loose class cohesion) based on the usage of direct or indirect attributes by public methods of the class. A many number of methods or measures proposed for In-house or OSS components are summarized below:-

Table 1:Metrics applicable to OSS and In-house Components with their gaps

Metric	Gap	Reference
Weighted Method Per class(WMC) (1994)	This method is based on counting Of number of methods in a class and no we scheme has been used For the prioritization of methods. This has been left as an implementation part	S.Chidamber and C.F.Kemere, “A Metric object oriented Design” IEEE Transact Software Engineering” Vol 20,1994
Depth of Inheritance (DIT)	This metric is somewhat to be uncertain because it is based on the hierarchy of the class to the root node and it may be as simple as well as complex for the different users. It also fails to satisfy the 4th property of Weyuker’s property of monotonicity.	S.Chidamber and C.F.Kemere, “A Metric Suite for object oriented Design”IEEE Transactions on Software Engineering”Vol 20,1994
Number of Children (NOC)	It is also based on the concept of inheritance and will give only the approximation values.	S.Chidamber and C.F.Kemere, “A Metric Suite for object oriented Design”IEEE Transactions on Software Engineering”Vol 20,1994
CBO	It is simply the count of number of classes to which it is coupled. It cannot be fruitful because generally we need the low coupling and high cohesion.	S.Chidamber and C.F.Kemere, “A Metric Suite for object oriented Design”IEEE Transactions on Software Engineering”Vol 20,1994
Class Request For Service (CQFS)	It is totally based on the interaction betwee classes during the execution of program.	Zaidman and Demeyer(2004) metric suite
Dynamic Coupling Metric(DCM) (for an object *P*)	It works only for a specified time period as delta t for a particular number of classes.	Hassoun *et al.*(2004&2005) metric suite
System Cohesion Metric	It does not follow the weyuker properties.	J.Chen et.al, “Complexity metric for component based software system”, International Journal of digital content technology and its applications, Vol 5, 2011.
Component Coupling (COC)	It is based on the internal structure of a component. Generally it works for simple components having a simple sharing of methods or attributes between components.	S. Patel and J.Kaur, “A Study of component based software system metrics”, pp. 824-828, ICCCA,2016.
Constraints Complexity(CTC)	Based on number of constraints on a component.	S. Patel and J.Kaur, “A Study of component based software system metrics”, pp. 824-828, ICCCA,2016.
Configuration Complexity (CFC)	Based on Configuration of components which will generally vary for every component.so a stability in results of complexity will not be there.	S. Patel and J.Kaur, “A Study of component based software system metrics”, pp. 824-828, ICCCA,2016.

Theoretical Summarization of measurement

In the following section we provide the basics of packages, classes and methods as well as their standard hierarchy have to be taken in to consideration.

Definition of packages, classes, methods

For the purpose of cohesion measurement, a package is de-fined as a set of elements (classes, interfaces or packages) and relations between elements. Generally a software is made up of thousand number of lines of code and for placing the appropriate interfaces and methods in to appropriate way packages are required.

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.

Package cohesion measurement approach using package cohesion component complexity metric (PC_3M)

Software metric plays a vital role in quantative assessment of any specific software development methodology and its impact on the maintenance of software. Component based software engineering (CBSE) is gaining substantial interest in the software engineering community. A lot of research efforts have been devoted to the coupling and cohesion metric for components in CBSE and a number of complexities metric have been proposed in the past. Complexity metric are calculated at package level by considering the relationship between the packages, classes and methods. Merely selection of less complex and reliable components is not sufficient but we have to select the optimal components for the software system including the factor of coupling and cohesion. We have proposed a new component coupling and component cohesion metric using the concept of component dependency to increase the reliability of software [10]**.**

Various no of steps used in calculating the package cohesion component complexity metric will be as follows:-

1) Considering the real data set.

2) Finding the parameters like DUM, IUM, and NDIUC.

3) Calculating the package cohesion complexity metric.

This value comes out to be as higher as possible.

Higher value of PC3M wills leads to low complexity of software. Parameters used in the metric are defined as follows:-

DUM:-It is the set of all direct connections between classes and methods.

NDIUC: - It is the number of used direct or indirect connections in the case study and it can be calculated as n(n-1)/2 where n is the total number of packages in the case study.The formula of n(n-1)/2 has been taken from the concept of TCC (Tight class cohesion) and LCC (Loose Class Cohesion) metric

PC3M:- Package cohesion component complexity metric.

PC3M =

$$\left\{ \begin{array}{ll} 0 & \text{if}(n=0) \\ (\text{DUM})/\text{NDIUC}) & \text{if } (n>1) \\ 1 & \text{if } (n=1) \end{array} \right\}$$

Where n is the number of elements(classes,methods)

Other cases:-If n=0, there is no element so no possible relation therefore computed value of PC_3M is also 0.

If n=1 means a single element is existing so the relation existing will also be single, hence the value of PC_3M is also 1

Illustration of package cohesion component complexity measurement(PC_3M) 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
      }
```

The computed value of cohesion measured by the PCoh Approach was 0.334 [22].Now the calculation by using the PC3M approach is as follows:-

DUM:-Since the package myshapes is having a single method so a single direct connection is existing hence the value of DUM is 1.

NDIUC=n (n-1)/2

Since no of packages used are 3 therefore n (n-1)/2 gives value 3 so

1/3=0.333

Therefore the value computed using PC3M Comes out to be less than PCOH.

Theoretical validation of package cohesion component complexity measure

The purpose of this section is to validate the package cohesion component complexity measure by using the four properties of Briand et al. [22].Many number of reviewers have used Weyuker's properties or framework for the validation of their proposed complexity metric. Some other evaluation frameworks such as Zuse framework [23] and Tian & Zelkowitz [24] axioms are also used for valida-tion of complexity measures.Briand et al. are the recent proposal for validating the metric.

Property 1:*Non-Negativity and Normalization*.

According to the above explanation given in the summarization of proposed measure the computed value of PC3M belongs to a specified interval of [0,10].Therefore the value computed by using above already defined measure will always be non-negative as well as normalized.

Property 2: *Null Value and Maximum Value*.

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 measure satisfies the property 2.

Property 3: *Monotonicity*.

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 DUM but simultaneously the value of IUDM get also increased as (IUDM=direct connection + indirect connection).hence adding more number of relations will not decrease the cohesion value and hence it satisfies the property of monotonicity.

Property 4: *Merging of Unconnected Packages*.

According to this property the merging of two unconnected packages should not increase the value of cohesion. The 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.

Experimental validation of proposed measure

In this section we are focussing on the experimental validation of package cohesion component complexity metric and showing the improved results of reliability with the impact on reusability. This would help us to evaluate the proposed package cohesion component complexity metric as a quality indicator.

Experiment analysis

The purpose of the following experimental study is to analyse experimentally the proposed metric for the purpose of evaluating the utility of the metric by comparing the values with the PCOh metric values. We used the template or guidelines provided for defining the measurement goas[25]. The measurement goal can defined as follows:

- Object of study: Number of packages and the relation between the packages, classes and methods.
- Purpose: analysis and comparison with the pcoh metric values.
- Quality focus: effort required to reuse a component (component Reusability).
- Viewpoint: software developer.
- Environment: open-source software projects developed in Java.

These above defined goals help to determine the type of data to be collected.

Empirical hypotheses

An empirical hypothesis is a statement believed to be true about the relation between one or more attributes of the object of study and the quality focus [26]. In this case, the hypotheses about the relationship between the packages, classes and methods and reusability of component (Quality focus)

The analysis is based on the hypotheses:

Hy_0: q=0 (Null hypothesis)—There is no significant cor-relation between the proposed package cohesion component complexity metric and component reusability.

Hy_1: q=0 (Alternative hypothesis)—There is significantcorrelation between the proposed package cohesion component complexity metric and component reusability.

Experimental environment

There are a various number of open source software projects available on the web 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 or-der 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].

Analysis methodology

A statistical analysis is performed to correlate the proposed package cohesion component complexity metric with component reusability for 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 usefulness [28]. Karl person Correlation coefficient isused for measuring the correlation.

This method assumes that there is a linear relationship between two variables and one of the variables is independent while the other is dependent. Karl Pearson's coefficient of correlation is given by [29]:

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

where $\overline{X_i}$ = ith value of $\overline{X}$ variable, Y_i = ith value of Y variable, X = mean of X, Y = mean of Y , n = number of pairs of observations of X and Y , σ_x = standard deviation of X, σ_y = standard deviation of Y .

The value of correlation coefficient (r) lies between −1 and +1 through 0, where 1 represents a perfect positive cor-relation between the variables; −1 denotes a perfect nega-tive correlation; and 0 indicates that there is no linear rela-tionship between the variables. The degree of the correlation is determined by the magnitude of the coefficient [22]

The ratings for the cohesion can be defined as:-

- 0 “minimum”
- 1.0 to 2.0 “just above the minimum”
- 2.0 to 0.5 “average”
- 4.0 to 5.0 “Best”
- 5.0 to 7.0 “optimum”.

Table 1 Description of packages used in the study[22]

Sr. No.	Project name	Package name	Size (LOC)	Total No. of classes
1	Byte Code Engineering Library (BCEL)	org.apache.bcel.verifier	12244	48
2		org.apache.bcel.verifier.exc	641	14
3		org.apache.bcel.verifier.statics	3425	9
4		org.apache.bcel.verifier.structurals	6289	14
5		org.apache.bcel.util	3906	20
6	Bean Scripting Framework (BSF)	org.apache.bsf.util.event	1790	21
7		org.apache.bsf.util.event.generator	1110	4
8		org.apache.bsf.util	5835	54
9		org.apache.bsf.util.type	239	2
10		org.apache.bsf.util.cf	570	2
11	Jakarta-ORO	org.apache.oro.io	494	4
12		org.apache.oro.text	14715	50
13		org.apache.oro.text.awk	3383	17
14		org.apache.oro.text.perl	1477	3
15		org.apache.oro.util	991	7
16	Element Construction Set (ECS)	org.apache.ecs.jsp	1834	15
17		org.apache.ecs.storage	452	3
18		org.apache.ecs.xml	786	3
19	XGen Source Code Generator	workzen.xgen.ant.legacy	1193	4
20		workzen.xgen.engine	426	2
21		workzen.xgen.loader	1009	7
22	Junit	junit.samples	541	2
23		junit.samples.money	390	4
24		junit.tests	1583	36
25		junit.tests.extensions	248	5

Table 2 Package cohesion measurement[22]

Sr. No.	Package name	No. of elements	No. of relations	Package cohesion metric (*PCoh*)
1	org.apache.bcel.verifier	14	56	0.30
2	org.apache.bcel.verifier.exc	14	10	0.15
3	org.apache.bcel.verifier.statics	9	1	0.013
4	org.apache.bcel.verifier.structurals	14	10	0.054
5	org.apache.bcel.util	20	10	0.026
6	org.apache.bsf.util.event	6	18	0.60
7	org.apache.bsf.util.event.generator	4	4	0.33
8	org.apache.bsf.util	20	30	0.078
9	org.apache.bsf.util.type	2	1	0.50
10	org.apache.bsf.util.cf	2	1	0.50
11	org.apache.oro.io	4	3	0.25
12	org.apache.oro.text	15	34	0.16
13	org.apache.oro.text.awk	17	41	0.15
14	org.apache.oro.text.perl	3	0	0
15	org.apache.oro.util	7	11	0.26
16	org.apache.ecs.jsp	15	14	0.067
17	org.apache.ecs.storage	3	3	0.50
18	org.apache.ecs.xml	3	2	0.33
19	workzen.xgen.ant.legacy	4	3	0.25
20	workzen.xgen.engine	2	1	0.50
21	workzen.xgen.loader	7	7	0.167
22	junit.samples	4	1	0.08
23	junit.samples.money	4	9	0.75
24	junit.tests	5	4	0.20
25	junit.tests.extensions	5	4	0.20

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 3:Calculated Values of Package Cohesion Component Complexity Metric

Sr. No.	Package Name	Total no of classes	No of elements	Fileted class(DUM)	No of relations (NDIUC)	PC3M
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	0	1	0
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

Analysis of experimental results

Table 3 shows the package name with the total number of classes, number of elements with their calculated filtered class and it is termed as DUM.The number of relations is termed as NDIUC and finally with all these the Package cohesion component complexity metric have been calculated.

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.

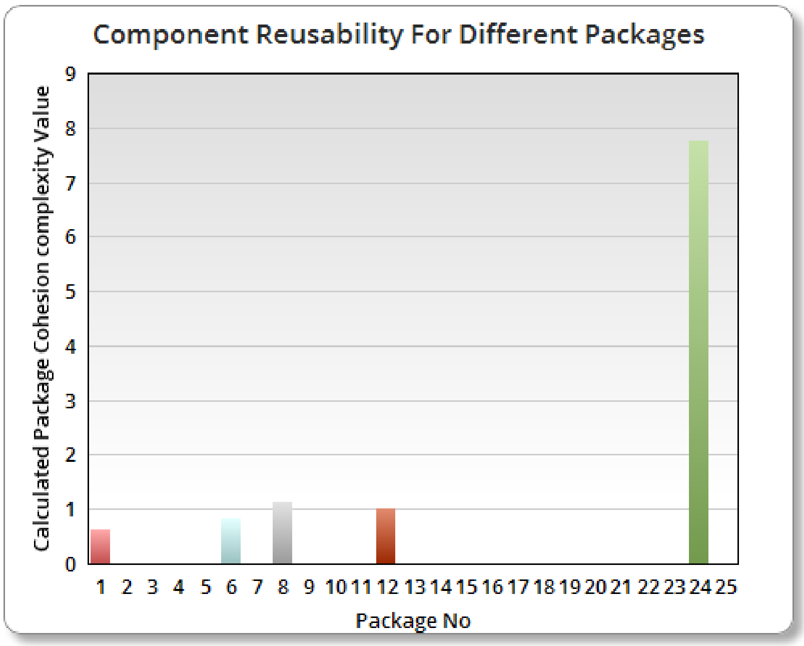
Table 4 Descriptive statistics of the analysed package cohesion component complexity measure (PC3M) metric

Statistical parameter	Package cohesion component complexity metric (*PC_3M*)
Maximum value	7.75
Minimum value	0
Median	1.02
Mean	0.05
Standard deviation	3.070

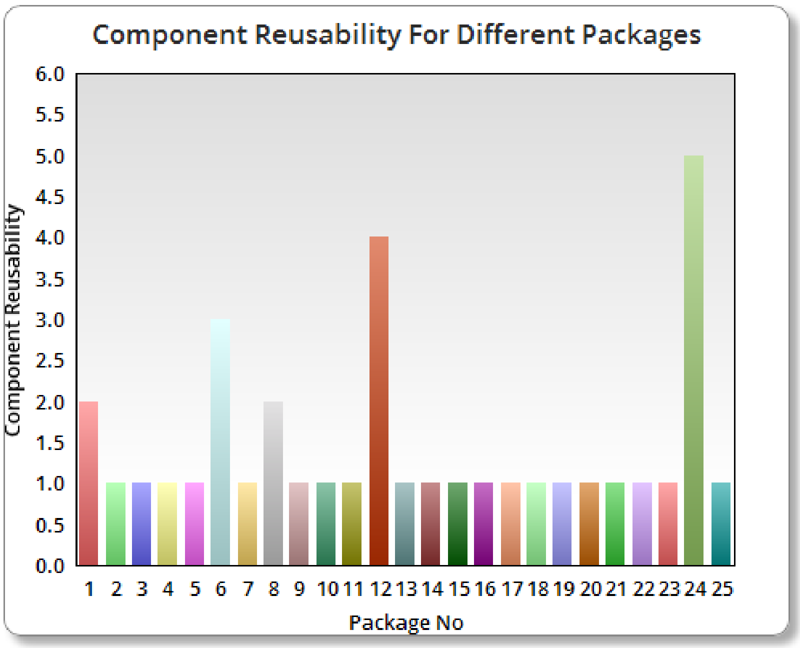
Table 4 Ratings of component reliability and reusability based on PC_3M

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

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.



Graph1: Calculated PC_3M value with package No



Graph2: Component Reusability on basis of Package cohesion component complexity

Comparison with the Correlation coefficient values of PC3M and other metrics

Parameters	PC_3M	LCOM	LCOM1	ICH	SCC
Correlation Coefficient	0.24	-0.32	-0.34	0.12	0.27
Significance value	0.05	0.05	0.05	0.05	0.05

From the above results it can be concluded that we can reject the null hypothesis and can trust on the alternative hypothesis. Hence there is a strong relation between the calculation of package cohesion component complexity metric and the component reusability.

Conclusions and future work

In this paper, we have analysed the package cohesion component complexity metric and he results are also compared with the previous Package cohesion metric. The PC3M is also validated using the recent used properties. The proposed measures are based on the direct and indirect relations between classes and methods. 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 rather than reusability, this metric can be used in order to establish the relations with some other factors in component based development environment like maintainability, adaptability.

References

[1] Basili VR (1997) Evolving and packaging reading technologies. J Syst Softw 38(1):3–12
[2] Corbi TA (1989) Program understanding: challenge for the 1990's. IBM Syst J 28(2):294–306
[3] Patel S, Chu WC, Baxter R (1992) A measure for composite mod-ule cohesion. In: Proceedings of the 14th international conference on software engineering, 1992, pp 38–48
[4] Martin R (2002) Agile software development, principles, patterns, and practices. Prentice-Hall, New York
[5] Dgfgfgf
[6] Telles M (2001) C# Black Book, The Coriolis Group, Nov 2001
[7] DeMarco T (1982) Controlling software projects. Yourdon Press, New York
[8] 2017 paper
[9] Briand L, Morasca S, Basili V (1996) Property-based software en-gineering measurement. IEEE Trans Softw Eng 22(1):68–86
[10] Bieman JM, Kang BK (1995) Cohesion and reuse in an object-oriented system. In: Proceedings of the symposium on software reusability (SSR'95), Seattle, WA, 1995, pp 259–262
[11] Bieman JM, Kang BK (1998) Measuring design-level cohesion. IEEE Trans Softw Eng 24(2):111–124
[12] Bieman JM, Ott LM (1994) Measuring functional cohesion. IEEE Trans Softw Eng 20(8):644–658
[13] Briand L, Morasca S, Basili V (1996) Property-based software en-gineering measurement. IEEE Trans Softw Eng 22(1):68–86
[14] Byte Code Engineering Library (BCEL) (2011) http://jakarta. apache.org/bcel/index.html
[15] Counsell S, Mendes E, Swift S (2002) Comprehension of object-oriented software cohesion: the empirical quagmire. In: Proceed-ings of the 10th international workshop on program comprehen-sion, 2002, pp 33–42
[16] Eder J, Kappel G, Schre M (1992) Coupling and cohesion in object-oriented systems. In: Proceedings of the conference on in-formation and knowledge. ACM Press, New York
[17] Emerson T (1984) A discriminant metric for module cohesion. In: Proceedings of the 7th international conference on software engineering (ICSE), 1984

[18] Chhillar R. S. and Kaijla P. 2011. A New Knot Model for Component Based Software Development". IJCSI, vol. 8, Issue 3, No. 2.
[19] Hitz M, 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, 1995, pp 25–27
[20] Ott L, Bieman JM, Kang B, Mehra B (1995) Developing measures of class cohesion for object-oriented software. In: Proceedings of the annual Oregon workshop on software metrics (AOWSM'95), 1995
[21] base Pper
[22] Zuse H (1991) Software complexity: measures and methods. de Gruyter, Amsterdam
[23] Tian J, Zelkowitz MV (1992) A formal program complexity model and its application. J Syst Softw 17:253–266
[24] Van Solingen R (2002) The goal/question/metric approach, ency-clopedia of software engineering—2 volume set, pp 578–583
[25] Briand L, Morasca S, Basili V (2002) An operational process for goal-driven definition of measures. IEEE Trans Softw Eng 28(12):1106–1125
[26] The Apache Jakarta Project (2011) http://jakarta.apache.org
[27] Li W, Henry S (1993) Object oriented metrics that predict main-tainability. J Syst Softw 23(2):111–122
[28] Kothari CR (2007) Research methodology: methods & techniques, revised second edn. New Age International, New Delhi, pp 139–141

AUTHOR PROFILE

Jyoti Sharma received her B.tech(IT) from Kurukshetra university and M.tech (IT)from BANASTHALI UNIVERSITY Jaipur. She is pursuing Ph.D in CSE from SRM UNIVERSITY, Haryana. Her research areas include software engineering,component based software engineering. She is having eight publications in international conferences and journals. She is a member of IEEE. She was a rank holder in B.tech in whole university.

Dr. Puneet Goswami is a Professor in Department of Computer Science and Engineering. He completed his Ph.D.(2013) & M.Tech(2004) with distinction in Computer Science & Engg. from 'A' Grade NAAC Accredited state technical University of Haryana Guru Jambheshwar University of Science & Technology, Hisar. He did his B.Tech from Maharishi Dayanand University, Rohtak. He has vast experience of 13 Years in teaching. His expert areas are Software Engineering, Big Data, Cloud Computing. He has Published a book titled "Big Data - A Driving Force for an Innovation" by Puneet Goswami, Somesh Kumar published on: 2016-03-14 bearing ISBN-13: 978-3-659-34099-4 ISBN-10: 3659340995 EAN: 9783659340994 published by Academic Publishing OmniScriptum GmbH & Co. KG Bahnhofstr. 28 66111 Saarbrucken Germany. He received an invitation from IBM to deliver a talk on the topic "Security in Cloud Reference Models and Secure Identity Management Mechanism" at 1st International IBM Cloud Academy Conference ICA CON 2012 at the IBM Employee Activity and Fitness Center, Research Triangle Park, North Carolina United States of America Organized by : The IBM Cloud Academy and IBM Centers for Advanced Studies (RTP, Chicago, Heritage Corridor, Tucson, Florida). He received Letter of Appreciation for completing challenging assignment of conducting first online Common Admission Test for Indian Institute of Management from 28th November to 8th December 2009 as Test Centre Administrator (TCA). 16 Scholars have been awarded M.Tech Degree in Computer Science and Engineering under his supervision. He is currently guiding seven Ph.D scholars .He has published 40 research papers in various international/National journals of repute. He is also an approved reviewer of various IEEE conferences. He is member of Board of studies of U.IET kurukshetra University kurukshetra. He has handled NMEICT Project by M.H.R.D Govt. Of India to conduct funded workshops on various engineering disciplines to enhance the skill set of faculties as well as students.