Cohesion Metric and Its Relation with Coupling: A Class Level Variable Assessment Approach

By Dr. M. V. Vijaya Saradhi, Dr. B. V. Ramana Murthy

Abstract - Cohesion Metrics is an important technical development which helps for the better Assessment Of class cohesion that is the measurement of relatedness among members In a class. The higher this relatedness is the best the performance will be. Hence this is an important feature of Object oriented systems. Present paper presents an advanced metrics. The pervious metrics have got few limitations. Whereas this advanced metrics considers few more characteristics of class Cohesion. This is based on common object parameters. Moreover this metric is statistically advanced and measures cohesion by observing the relationship between cohesion Metric and Coupling. This is cohesion measurement tool for Java and it is tested on several systems. These systems which are used for the experiment are deferent in size and domain. This test proved that this extended metrics captures other pairs of relatedness among members in a class and also the correlation between cohesion Metric and coupling.

Keywords : Metrics, Cohesion, LCOM, CBO, MIC, DCD, DCDE, DCI, TCC, LCC, MIC, DCIE, Coupling, Class level variables, Spearman correlation, MPC, CLC, OLC.

GJRE Classification : FOR Code: 050299

Strictly as per the compliance and regulations of :

© 2011 Dr. M. V. Vijaya Saradhi, Dr. B. V. Ramana Murthy. This is a research/review paper, distributed under the terms of the Creative Commons Attribution-Noncommercial 3.0 Unported License http://creativecommons.org/licenses/by-nc/3.0/), permitting all non commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Cohesion Metric and Its Relation with Coupling: A Class Level Variable Assessment Approach

Dr.M.V.Vijaya Saradhi*, Dr.B.V.Ramana Murthy²

Abstract - Cohesion Metrics is an important technical development which helps for the better assessment of class cohesion that is the measurement of relatedness among members in a class. The higher this relatedness is the better the performance will be. Hence this is an important feature of Object-oriented systems. Present paper presents an advanced metrics. The previous metrics have got few limitations. Whereas this advanced metrics considers few more characteristics of class Cohesion. This is based on common object parameters. Moreover this metric is statistically advanced and measures cohesion by observing the relationship between cohesion Metric and Coupling. This is cohesion measurement tool for Java and it is tested on several systems. These systems which are used for the experiment are different in size and domain. This test proved that this extended metrics captures other pairs of relatedness among members in a class and also the correlation between cohesion Metric and coupling.

Keywords : Metrics, Cohesion, LCOM, CBO, MIC, DCD, DCDE, DCI, TCC, LCC, MIC, DCIE, Coupling, Class level variables, Spearman correlation, MPC, CLC, OLC.

1. INTRODUCTION

In Software Engineering there is a raising importance to Metrics and cohesion which are interlinked [28, 27]. Metrics are very much useful in assessing several software characteristics such as complexity, cohesion, coupling and size. Cohesion helps for the better performance and metrics to decide the level of cohesion. So both are equally important in the field of software development. Hence research is going on to improve the cohesion as well as metrics which is a tool to measure it. Cohesion can be defined as the degree of relatedness among elements in a component. It was first introduced and utilized by Yourdon and Constantine in the context of traditional applications. They used it as a tool to estimate level of functional relationships of the elements in a module [30]. These modules are structured for different uses. Class cohesion is a very important feature of object-oriented software. This feature of software is of great help to software developers and managers to improve the software quality during the development process.

There are three different types of cohesions. Functional, sequential and coincidental etc [19] is few among them. This feature of software is important because of its wide range uses. If the component has good range of cohesiveness among its members, it can be reused and maintained well [25, 7, 9, 13]. With the help of cohesion one can evaluate the structure quality of class. A class which has good cohesiveness cannot be broken easily. It can be used to find out the poorly structured classes. In addition while designing a class cohesion is of great assistance to bring out the best [24]. As a reference we can observe Grady Booch views. He describes that high functional cohesion can be achieved if the elements of a component have good cohesiveness among them which provides some well bounded behavior [8]. This can be brought with single logical function. If all the parts of a class contribute to this single logical function high degree of cohesiveness can be achieved. In contrast if the members are desperate and non related then the coherence will be very low. Since cohesion has such a large scale importance, much number of metrics was proposed to estimate it in object-oriented systems. Most of these metrics have been experimented and widely discussed in literature [19, 12, 16, 18, 11, 5]. From the above paragraph one can understand that class cohesion is a very important feature of object-oriented systems. Many successful metrics have been elaborated and categorized in [10]. These metrics estimate the cohesiveness according to them relatedness among the members of the class. They count two features: first one is the number of instance variable used by methods and second is the number of method pairs that share instance variables. Though many metrics were proposed in literature they were not total successful in finding out the cohesiveness of classes [22, 13, and 3]. There are some basic reasons for this failure. Some of them are that they do not undertake few features of classes that are the size of cohesive components and relatedness among elements as said in [13]. Few other serious drawbacks which above stated metrics are that they are based only on few categories like instance variable and number of method pairs as stated in [23]. This often leads to wrong estimation of the cohesiveness among members in a component. So the previous metrics face a serious problem when the systems work in functional relationship. In this category, cohesiveness can not be
decided by the above said connections but has to be
done with the help of the relationships that may exist
among methods. If the same old metrics is followed
many features of class cohesion will not represented.
Hence we believe that class cohesion will not be exact if
it does not go beyond above cited categories. Research
on source code on several systems tells us that several
methods functionally attached even without sharing any
instance variable and these can’t be divided into
different classes. As such the focus is to be extended by
taking into account different ways of estimating class
cohesion and should not be restricted to any two. First
development of the metrics is that connections among
class methods will be considered [5, 6]. These systems
help to find out much number of pairs of related
methods which are not found by previous cohesion
metrics. This criterion proved successful when it was
tested on several Java systems. In these experiments
they gave correct statistical information. In these last
years many such developments were brought in. of
them [17], concentrates on tradition of maintainability
[Zh03] and [1] on the intimation of mistakes, and [23]
on examine the relationship between cohesion and
coupling in the one hand and the relationship between
cohesion and changeability in the other hand. The area
has a raising importance in these last years.

Till now cohesion metrics were limited to object
oriented systems but we used this information for other
systems also [5, 6]. The previous metrics were based
only on instant variables and number of methods pairs.
But this information is very primitive to depend on. With
this one cannot give good cohesion results. As such
research was undertaken to find out other categories
which are more authentic and will be helpful to give
exact cohesive results. This paper gives an extension of
two methods that is DCD and DCI which was proposed
in [5, 6]. A new criterion of common object parameters
was introduced to calculate cohesion levels. This
criterion tells us that two methods of a given class can
very well share same object passage in a parameter
without being correlated. It does not need to share a
method or an instance variable to get connected. So
depending on instance variable take us wrong, where as
this object passage can present authentic results.
Further more it was discovered that in the object context
objects themselves collaborate to accomplish a given
task. As such certain design principles [24] like design
patterns among others and classes play an important
role in successful completion of a given job. This
collaboration can be located at two levels. One at the
group of objects belonging to different classes, second
at the collaboration between groups of methods with in
a unique given class. This last kind of collaboration can
be observed among other things also. These are used in
the form of instance variable or passes as arguments at
the method level, public in particular. In this kind of
collaboration cohesion helps to assign responsibilities to
classes [24] in a cohesive manner. Form the above
conclusions and according to the experiments done
since 2003, this new category to estimate cohesion
levels is more dependable. This is proved after
conducting many experiments on systems. These
clearly show that the extended cohesion metrics based
on the addition of the proposed category captured more
pairs connected methods than the old metrics DCD and
DCI did. These experiments that were done on several
systems gave correct, authentic, statistical results.

Software Engineering developers state that
there is a correlation between cohesion and coupling.
They state that if the cohesion is high, coupling will be
low and vice versa [24, 28, 27]. But this notion was not
proved by any empirical work. Many experiments were
done to bring out the realities and they could only
present the necessity for a refined cohesion metrics.
They failed because of the limitations of the previous
metrics [23]. This paper presents such an extended
cohesion metrics. This technique is tested to find out the
truth in the relationship between extended cohesion and
coupling. Here it was decided to use both the old and
new cohesion metrics. The experiment shows that there
is a significant correlation between our cohesion metrics
and the considered coupling metrics [CBO – Coupling
between objects] of Chidamber et al [14, 15]. But the
correlation degrees between them varied much when
they are presented by the considered cohesion metrics.
The empirical experiments as well as the obtained
results are discussed in section [7]. Our final goal is to
validate the new cohesion metrics as a good indicator
for changeability, testability and for many more things.
This will be dealt seriously in the future research.

The following paper is arranged in the following
way: Section 2 presents an overview of major class
cohesion metrics. Section 3 gives the idea of coupling
between objects and few important coupling metrics.
Section 4 gives some related work which focuses the
relatedness of object-oriented metrics and few quality
characteristics. Section 5 redefines class cohesion that
was proposed depending on the new criterion that we
introduced in this paper. Section 6 gives the first step of
the experiment that was done (statistic test). Section 7
gives the empirical investigation that we have done to
find out the relationship between cohesion and
coupling. Finally, conclusions and future work ideas are
presented in section 8.

II. CLASS COHESION METRICS

Classes are that primary units of object oriented
software. In these classes we find cohesion. It is an
important feature in software design. The higher the
cohesion better the performance will be. A class will
have best cohesiveness as stated in [13] if large number
of its instance variables are used by a method (LCOM5
[19], Coh [10]), or a larger number of methods pairs
share instance variables (LCOM1 [14], LCOM2 [15],
LCOM3 [25], LCOM4 [21], Co [21], TCC and LCC [7],
DC [4]). Other than the above two, sometimes these

© 2011 Global Journals Inc. (US)
metrics also observe the relatedness between the methods to assess cohesion. To get such good cohesion software developers struggle hard at the design phase of classes. If the modeling of these classes is not at its best then cohesion will automatically go down. Hence after designing the classes one would like to assess class cohesion. To assess them many metrics have been proposed in literature. Different authors have defined class cohesion by proposing their cohesion metrics. These cohesion metrics have been presented in detail and are categorized in [10]. One such cohesion metric is LCOM. It is lack of cohesion in methods. It is a metric which is defined by Chidambar and Kemerer [14,15,16]. This metric stands as a role model for many other proposed cohesion metrics. In addition to other proposed metrics, few others tried to redefine LCOM itself. All this cohesion metrics have come into literature to find out the class cohesion in object oriented systems.

a) Coupling Between Classes

Cohesion metrics measure cohesion between members. Whereas coupling measures the strength of a connection between two modules. Stevens & al [29] explain coupling as one which assesses the strength of the association which is formed by the relatedness between two modules. Coupling between classes helps to assess in which proportion an entity uses other entities. There are both positive and negative effects of the coupling. To speak of the positive low coupling between components helps to minimize interdependencies and gives a chance for evolution [24, 28, 27]. If the modules are structured with low coupling then the complexity of a system can also be minimized. On the negative side because of high coupling module becomes complex. Because of this complexity module will be tough to understand, tough to detect and correct errors, and to change that module. After analyzing these effects one can understand that low coupling is preferable. So it is always encouraged by software engineers. After much research it was discovered that with the help of these coupling metrics the maintainability of the oo systems can be easily imagine. Moreover there are few empirical insufficiencies due which there is lot of importance to coupling metrics for the preservation in maintainability [17]. Among coupling metrics, we cite CBO (Coupling Between Objects) of Chidamber and Kemerer [15], MPC (Message-Passing Coupling) and DAC (Data Abstraction Coupling) of Li and Henry [25, 26] or OLC (Object Level Coupling) and CLC (Class Level Coupling) of Hitz and Montazeri [21]. Brian & al. counted 23 coupling metrics [9]. For this research work CBO was used which is proposed by [15]. Largely known as a good coupling metric between classes. In our future work, we plan on extending our study to integrate other coupling metrics.

III. Related Work

During the research of the last three decades many software metrics like coupling, cohesion, and complexity were proposed to calculate certain aspects like maintainability, testability, changeability and many more things. But in finding out the quality these were not totally successful. Few reasons are that they are to some extent based on the little understanding of the empirical hypothesis. In addition to it all these proposed metrics can be used to determine any aspect of quality. There is no particular division that this metrics is for the assessment of this quality. As such there is no information about which metric is most suitable to assess a quality. This is the second major problem. Through the research it was also found that of all the metrics only six are efficient and can present sufficient information to depend on. Whereas all the remaining metrics can’t give any extra information and they just correspond to subsets of the retained metrics. These drawbacks were discussed by many engineers. Dag & al opines in [17] about the prediction of maintainability. He argues that because of the empirical insufficiencies that these metrics have got the assessment will not be very much dependable. Of all the research papers Dagpinar & al [17] paper presents in new development. In this paper they opines that inheritance cohesion and indirect exportation coupling are not the right factors by which maintainability can be measured well. They advise taking to consideration metrics of size and direct coupling importation which can give good results. A lot of research was undertaken to find out the correlation among the proposed coupling metrics and how much they are prone to fault results. One such research was done by Aggar & al [1]. In prediction model of [2] it was clearly proved that these metrics are very much prone to fault reasons. Zhou & al [31] also undertake similar work to find out the relationship among design metrics (CBO, WMC, RFC, LCOM etc) and fault proneness when taking fault severity into account. This was understood after conducting a thorough research on many number of oo coupling metrics [1]. This study focuses to find out the best methods according to the given data. After all the research about the relationship between Coupling and cohesion leads us to confusion that their may exist a connection between cohesion and for example maintainability, testability as well as fault proneness. Any how a lot more is necessary to find out the direct relationship that may exist between cohesion and above said attributes. This final aspect will be the subject of further research and is out of discussion of the present paper.
IV. Class Cohesion Assessments: A New Measure

Class cohesion at the beginning of this paper is defined as the relative number of related members in a class. This definition is redefined twice in this paper so as to get best methodology which can give authentic assessments. As a first step, two more strategies were added to the relative number. First one is the extension of the methods invocation criteria and indirect utilization of the characteristics explained by Bienan & al in [17]. This idea was extended to the methods invocation criterion as well. After defining the new methodology was tested on several systems [5,6]. This shows a lot of improvement in assessment than the first noted procedure. From the results one can observe that new criteria and the extension of the original criteria are capable of finding out more pairs of connected methods which were not found by the old methods. To come to this conclusion the procedure was tested on several systems. These experiments gave a chance to observe the code of some program. With this code observation and from the obtained results it was found that methods of a class may be functionally, related in other ways. In addition some facts about attributes were also found. From the experiment it was known that attributes, on which first development is dependent on, are not unique to any method. These attributes in reality are reference attribute. Such a one which is not unique but shared is used to assess class cohesion. Many systems were analyzed to come to the above conclusion and observations say that more than 20% of the attributes were reference attributes. This is very much possible oo systems because classes collaborate in accordance with the respective responsibilities so as to finish a given task. Reference attributes are utilized to confirm the needed visibility between objects [24]. Because of these drawbacks we tend to improve this second definition also. We tried to bringing in criteria which will not be primitive, shared but will be more authentic. For this a new criteria that is common objects parameter is introduced. In the following page we explain this and the metrics which work with this new methodology. Our first and second procedures to assess cohesion were already talked about and also tested in the previous papers [5,6]. This newly introduced criterion is the one which will be prominently discussed in this paper. Both these ways have got many similarities. This new way to assess class cohesion is very much dependent on different connections that are present between its methods. All the three proposed criterions: Attributes Usage Criterion, Methods Invocation Criterion, and Common Objects Parameters will be utilized to find out the functional cohesion in a class. Class cohesion can be said as the connectedness of public methods of a class, with the help of functionalities utilized by its clients. The others methods of the class are included indirectly through the public methods.

a) Attributes Usage Criterion (UC)

Let us take a class C. Let \( A = \{A_1, A_2, \ldots, A_a\} \) be the group of its characteristics and \( SPM = \{M_1, M_2, \ldots, M_n\} \) be the group of its public methods. Let \( UCM_i \) be the group of all the characteristics used directly or indirectly by the public method \( Mi \). A characteristic is used directly by a method \( Mi \), if the characteristic shown in the body of the method \( Mi \). The characteristic is indirectly used by the method \( Mi \), if it is used directly by another method of the class that is implored directly or indirectly by \( Mi \). There are \( n \) sets \( UCM_1, UCM_2, \ldots, UCM_n \). Two public methods \( M_i \) and \( M_j \) are indirectly called by the public relation if \( UCM_i \cap UCM_j \neq \emptyset \). It shows that there is at least one characteristic shared (directly or indirectly) by the two methods.

b) Methods Invocation Criterion (MIC)

Let us take a class C. Let \( SPM = \{M_1, M_2, \ldots, M_n\} \) be the group of its public methods and \( PRM = \{I_1, I_2, \ldots, I_k\} \) be the group of its other (private and protected) methods. Let \( SPMM_i \) be the group of all the public methods of the class C, which are implored directly or indirectly by the public method \( Mi \). A public method \( Mi \) is called directly by a public method \( Mi \), if \( Mi \) is seen in the body of \( Mi \). A public method \( M_j \) is indirectly called by a public method \( Mi \), if it is called directly by another method of the class C that is implored directly or indirectly by \( Mi \). There are \( n \) sets \( SPMM_1, SPMM_2, \ldots, SPMM_n \). Let \( PRM_Mi \) be the group of all the other methods (private and protected) of the class C, which are implored directly or indirectly by the public method \( Mi \). There are \( n \) sets \( PRM_M1, PRM_M2, \ldots, PRM_Mn \). Let \( MIC_Mi = PRM_Mi \cup SPMM_i \) be the group of all the methods of the class C, which are implored by the public method \( Mi \). There are \( n \) sets \( MIC_M1, MIC_M2, \ldots, MIC_Mn \). Two public methods \( M_i \) and \( M_j \) are directly connected by the MIC relation if \( MIC_Mi \cap MIC_Mj \neq \emptyset \). We also take it into account that \( M_i \) and \( M_j \) are directly related if \( M_j \in MIC_Mi \) or \( M_i \in MIC_Mj \).

c) Class level variables (CO)

Let us consider a class C. Let \( SPM = \{M_1, M_2, \ldots, M_n\} \) be the group of its public methods. Let \( UCOMi \) be the group of all the parameters (of object type) of the method \( Mi \). There are \( n \) sets \( UCOM1, UCOM2, \ldots, UCOMn \). Two public methods \( M_i \) and \( M_j \) are directly related by the UCO relation if \( UCOM_i \cap UCOM_j \neq \emptyset \). From the above we understand that there is at least one parameter of object type that is utilized by the two methods.

d) Cohesion based on the direct relation

Two public methods \( M_i \) and \( M_j \) may be directly interlinked in different ways: they share at least one instance variable in common (UC relation), or get connected at least with another method of the same class (MIC relation), or share at least one object passed as argument (CO relation). In this context, the two methods may be directly interlinked by one or more
criteria. It shows that the two methods are directly
interlinked if:  \( U_{M_i} \cap U_{M_j} \neq \emptyset \) or \( IM_{M_i} \cap IM_{M_j} \neq \emptyset \) or
\( UCO_{M_i} \cap UCO_{M_j} \neq \emptyset \). Let us consider a class \( C \) with \( SPM = \{ M_1, M_2, \ldots, M_n \} \) in character are
directly connected. Let \( ED \) be the number of edges in the graph \( GD \). The
level of cohesion in the class \( C \) is dependent on the direct
connection between its public methods is explained as: \( DC \) the group of its public methods. The
highest number of public methods pairs, is \( n * (n - 1) / 2 \). Let us take an undirected
graph \( GD \), in which vertices are the public methods of the class \( C \), and there is an
edge between two vertices if the methods which are
equal \( DC_{DE} = |ED| / [n * (n - 1) / 2] [0,1] \). \( DC_{DE} \) (as an
extension of \( DC_D \) [5, 6]) presents the percentage of
public methods pairs, which are directly (as defined
below) connected. The Lack of Cohesion in the Class
\( (LCC_{DE}) \) is than given by: \( LCC_{DE} = 1 - DC_{DE} \in [0,1] \).

e) Cohesion based on the indirect relation

Two public methods \( M_i \) and \( M_j \) can be indirectly
connected if they are directly or indirectly related to a
method \( M_k \). The indirect relation, brought in by Bieman
and Kang in [7], is the transitive closure of the direct
relation. We use this idea in our method to mark the
indirect related methods. Let us take now an undirected
graph \( GI \), where the vertices are the public methods of
the class \( C \), and there is an edge between two vertices if
the methods are directly or indirectly connected
(transitive closure of the graph \( GD \)). Let \( EI \) be the
number of edges in the graph \( GI \). The degree of cohesion in the class \( C \) in this case (direct and indirect
relations) is said as: \( DC_{IE} = |EI| / [n * (n - 1) / 2] \in [0,1] \). \( DC_{IE} \) (as an
extension of \( DC_{I} \) [5, 6]) presents the percentage of
public methods pairs, which are directly or indirectly related. The Lack of Cohesion in the Class
\( (LCC_{IE}) \) is than given by: \( LCC_{IE} = 1 - DC_{IE} \in [0,1] \).

V. Experimental Study

Several systems were downloaded from the
web to experiment on the new criterion. The goal was to
achieve significant and general results. To collect the
significant data was the main goal of these experiments.
Hence many number of Java classes from different
systems are taken. Through this experiment it was
explored if the proposed criterion is statistically
significant before more investigation. We extended the
cohesion measurement tool (in Java) for Java programs,
that we developed for [6], to automate the computation
of our metrics (DCD, DCDE, DCI and DCIE). Many
classes in the chosen systems have only one method or
do not have any methods. These classes were taken as
special classes and have not used for our
measurements. All abstract classes are also not used.
Overloaded methods within the same class were taken
as one method. In addition to it, all special methods
(constructors, destructors) were not used. We gathered
the values for all the selected metrics from the test
systems. For each metric, we calculated some
descriptive statistics (minimum, maximum, mean,
median, and standard deviation).

VI. SELECTED SYSTEMS

The experiment concerned more than 800
classes. The followed methodology and the obtained
results are presented in the following sections. The
selected systems are:

- **System1**: JIU0.10 (Java Imaging Utilities) is a library
  in Java for the change, the edition, the analysis and the
  backup of pixels of image files (http://sourceforge.net/projects/jiu). This system
  consists of 180 classes.
- **System2**: JIU0.11 (Java Imaging Utilities) is an
  improvement of the first system (http://sourceforge.net/projects/jiu) and consists of 191
classes.
- **System3**: FujabaUML is a software development tool
  which helps for the easy betterment of UML and the
  progress with Java by adding plug-ins (http://www.fujaba.de). This system consists of 186
classes.
- **System4**: Wbemservices is a Java open source
  implementation of Web Based Enterprise Management
  (WBEM) for commercial and non commercial
  applications. It is compiled of API, of servers, client
  applications and tools
  (http://wbemservices.sourceforge.net/). It contains 463
classes.

<table>
<thead>
<tr>
<th>Systems</th>
<th>Des. Stat</th>
<th>DC_D</th>
<th>DC_DE</th>
<th>DC_I</th>
<th>DC_IE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jiu1</td>
<td>Mean</td>
<td>0.16027</td>
<td>0.17384</td>
<td>0.1922</td>
<td>0.2178</td>
</tr>
<tr>
<td></td>
<td>Sdt.dev</td>
<td>0.13686</td>
<td>0.1378</td>
<td>0.1638</td>
<td>0.2178</td>
</tr>
<tr>
<td>Jiu2</td>
<td>Mean</td>
<td>0.2497</td>
<td>0.2635</td>
<td>0.3102</td>
<td>0.3350</td>
</tr>
<tr>
<td></td>
<td>Sdt.dev</td>
<td>0.16466</td>
<td>0.1714</td>
<td>0.2292</td>
<td>0.2246</td>
</tr>
<tr>
<td>Fujaba</td>
<td>Mean</td>
<td>0.01597</td>
<td>0.05244</td>
<td>0.0207</td>
<td>0.0656</td>
</tr>
<tr>
<td></td>
<td>Sdt.dev</td>
<td>0.01479</td>
<td>0.05861</td>
<td>0.0201</td>
<td>0.0739</td>
</tr>
<tr>
<td>WBEM</td>
<td>Mean</td>
<td>0.08138</td>
<td>0.2286</td>
<td>0.1013</td>
<td>0.2747</td>
</tr>
<tr>
<td></td>
<td>Sdt.dev</td>
<td>0.14164</td>
<td>0.2051</td>
<td>0.1678</td>
<td>0.2332</td>
</tr>
</tbody>
</table>

Table 1: Average values of cohesion.

VII. RESULTS

We assessed cohesion values for the 4 chosen
systems. Table 1 gives the mean values of the metric for
the chosen system. The results that we have got for
DCDE et DCIE show clearly that they find more pairs of
connected methods than DCD et DCI did.
From the above figures we can come to a conclusion that DCDE and DCIE are able to find out many other details of attributes of classes which are not found by other metrics. Through this research we would like to prove the importance of new criteria. Hence we are not going to evaluate the cohesion values of the selected systems. From the above given statistics in table 1, it can be said that these systems don’t have cohesiveness.

VIII. VALIDATION OF THE NEW CRITERION

The goal of this chapter is to check the effects of DCD and DCDE on one side and the effects of DCI and DCIE on the other. The goal of this comparison is to find out if there is any difference brought by the introduced new criteria. Through this we would like to prove that DCDE and DCIE are much preferable to DCD and DCI. Because DCDE and DCIE help to find out more pairs of related methods. To prove our above said assumption we have under taken one statistical test: the PAIRED t-TEST [20]:

Let \( \mu_1 \) be the mean value of DCDE (or DCIE) and \( \mu_2 \) be the mean value of DCD (or DCI).

Below we give two statistical hypotheses :

- \( H_0 : \mu_1 = \mu_2 \) The metrics are equivalent.
- \( H_1 : \mu_1 > \mu_2 \) DCDE (or DCIE) is more significant than DCD (or DCI).

Let Diff be the value of \((\mu_1 - \mu_2)\). The above test is equivalent to:

- \( H_0 : \text{Diff} = 0 \).
- \( H_1 : \text{Diff} > 0 \).

The test statistic is: \( Z = d / [S_d / \sqrt{N}] \)

With \( d \) : the mean value of sample Diff
\( S_d \) : the standard deviation of sample Diff and
\( N \) : the number of classes in sample Diff.

Tables 2 and 3 present respectively the comparison between DCD and DCDE on one side and DCI and DCIE on the other.

**Table 2: Comparison between DCD and DCDE**

<table>
<thead>
<tr>
<th>Systems</th>
<th>Des. Stat</th>
<th>DCD</th>
<th>DCDE</th>
<th>Diff</th>
<th>Z</th>
<th>Z( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jiu1</td>
<td>Mean</td>
<td>0.16027</td>
<td>0.17384</td>
<td>0.01356</td>
<td>1.799</td>
<td>1.645</td>
</tr>
<tr>
<td></td>
<td>Sdt. dev</td>
<td>0.13686</td>
<td>0.1378</td>
<td>0.01685</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jiu2</td>
<td>Mean</td>
<td>0.2497</td>
<td>0.2635</td>
<td>0.0228</td>
<td>2.4635</td>
<td>1.645</td>
</tr>
<tr>
<td></td>
<td>Sdt. dev</td>
<td>0.16466</td>
<td>0.1714</td>
<td>0.0207</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fujaba</td>
<td>Mean</td>
<td>0.01597</td>
<td>0.05244</td>
<td>0.03646</td>
<td>2.6547</td>
<td>1.645</td>
</tr>
<tr>
<td></td>
<td>Sdt. dev</td>
<td>0.01479</td>
<td>0.05861</td>
<td>0.05663</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WBEM</td>
<td>Mean</td>
<td>0.08138</td>
<td>0.2286</td>
<td>0.1472</td>
<td>4.7917</td>
<td>1.645</td>
</tr>
<tr>
<td></td>
<td>Sdt. dev</td>
<td>0.14164</td>
<td>0.2051</td>
<td>0.1869</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

They clearly show that, for the many tested systems \( Z\alpha \) lower than \( Z \). The systems for which \( Z\alpha \) is higher than \( Z \) are the systems for which \( N \) is low. Through out the world, the results show that DCDE (or
DCIE) is preferable than DCD (or DCI). This statistical validation shows the applicability of the new cohesion criterion for finding new pairs of connected methods. The results that we have got prove that the extended cohesion metrics, based on the newly introduced criteria, find more pairs of connected methods than metrics DCD and DCI.

IX. THE RELATIONSHIP BETWEEN EXTENDED COHESION ASSESSMENT AND COUPLING

To validate our metrics we went for further experimentation. Software developers believe that cohesion and coupling are correlated. Though not proved it is said that coupling will be low when cohesion is high and vice versa [24, 28, 27]. Using our new criteria of cohesion we tried to know the facts about this belief. If the facts about this can be brought out, we can prove that new metrics on the new criteria is most successful way of assessment. Through these experiments we can bring out the relationship that the metrics can directly have with high level quality attributes like testability, changeability and maintainability. But this is a very beginning stage and no conclusions can be brought. We need more research to confirm the above relationship.

An empirical study

The experiment we performed considered six systems that vary in size (number of classes) and domain. The selected systems are (more than 500 classes):

- System 1: Gnujsp 1.0.1, GNUJSP is a free implementation of Java Server Pages of Sun (http://klomp.org/gnujsp). This system contains 56 classes.
- System 2: JIU 0.12, JIU (Java Imaging Utilities) is a library in Java for loading, editing, analyzing and saving pixels in image files (http://sourceforge.net/projects/jiu). This system has 77 classes.
- System 3: fujabaUml.4, FujabaUML is a software development tool allowing the easy extension of UML and Java development with the use of plug-ins (http://www.fujaba.de). This system contains 60 classes.
- System 4: jexcelapi 2.6, JExcelApi is a Java library that grants the possibility of reading, writing and modifying Microsoft Excel Worksheets (http://sourceforge.net/projects/jexcelapi). It contains 110 classes.
- System 5: moneyjar 0.8, Moneyjar is a Java library for financial applications. It simplifies treasury management, currency exchange, tax calculations and invoice management (http://sourceforge.net/projects/moneyjar). It contains 20 classes.
- System 6: wbemservices 1.0.0, Wbemservices is an open source Java implementation of Web Based Enterprise Management (WBEM) for commercial and non commercial applications. It is a project composed of APIs, of servers, of client applications and of tools (http://wbemservices.sourceforge.net/). This system contains 180 classes.

Experimental Process: First phase

We started our experiment to find out the relationship between cohesion and coupling. Of all the selective systems six were chosen and taken for the experiments. From these data is collected about the four cohesion metrics and also about the CBO metrics. The study conducted with this data proves that there is a definite correlation between cohesion and coupling. Here after it is not a belief but fact that when cohesion is high coupling will be low and inverse is also true.

Experimental Process: Second phase

In this second step we would like to explain how we come to the above conclusion. From the following results we can prove the hypothesis that there exists a relationship between coupling and cohesion. To prove the above four cohesion metrics: DCI, DCD, DCDE and DCIE and for coupling CBO metric are undertaken. As a first step data on the selected metrics from each of the considered systems is collected. Later to find out the relationship Spearman coefficient was used. This experiment is important because it proves the above said belief. This test is well suited since the dependence seems to be non linear as stated to the previous graphs. Studies of the data sets are done by calculating the Spearman dependence coefficients for each pair of metrics (a metric of cohesion, CBO). The Spearman statistic is based on ranks of the observations. The value of the Spearman statistic is a number between -1 and 1, -1 being a perfect negative dependence and +1 a perfect positive dependence.

Results

X. REGRESSION STUDY

Main aim of this study is to find out if there is any linear connection between cohesion metrics and coupling. For this a regression study was done between coupling and under different cohesion metrics. As a first step cohesion metric connected to the retained coupling.

Later a regression evaluation between two variables was done. Below are some terms utilized in this section of the paper.

- Regression model: It is the regression model used. DCDE, DCIE, DCD, DCI are the variables which are not dependent and coupling metric CBO is not independent
- Dependant variable: A random variable to predict;
- Independent variable: A predictive variable;
- R^2 (r-square): The percentage of change in the dependent variable described by the independent variables in the regression model for the given example of the population.
To analyze some other variant of this relatedness between the metrics of cohesion and the coupling metric, the logarithm of the coupling value was explained. To get this value a regression between this logarithm and the cohesion is undertaken. The outcome of the above is given in table 5.

For this first experiment we have utilized $R^2$ statistic through this we have tried to find out the areas that connect coupling and cohesion. To find out this, values of system JIU are used. For this DCDE and DCIE values are 0.0228 and 0.0267 respectively. These values present the variance of coupling brought in by the cohesion metrics, which can be said as 2.28% and 2.67% in percentages.

Table 4 : Values of $R^2$ in the different systems.

<table>
<thead>
<tr>
<th>System</th>
<th>Cohesion Metric</th>
<th>$R^2$ vs Coupling</th>
</tr>
</thead>
<tbody>
<tr>
<td>FujabaUml</td>
<td>$DC_{DE}$</td>
<td>0.0118</td>
</tr>
<tr>
<td></td>
<td>$DC_{IE}$</td>
<td>0.0081</td>
</tr>
<tr>
<td></td>
<td>$DC_D$</td>
<td>0.0081</td>
</tr>
<tr>
<td></td>
<td>$DC_I$</td>
<td>0.0054</td>
</tr>
<tr>
<td>Gnujsp</td>
<td>$DC_{DE}$</td>
<td>0.2835</td>
</tr>
<tr>
<td></td>
<td>$DC_{IE}$</td>
<td>0.2676</td>
</tr>
<tr>
<td></td>
<td>$DC_D$</td>
<td>0.4657</td>
</tr>
<tr>
<td></td>
<td>$DC_I$</td>
<td>0.4506</td>
</tr>
<tr>
<td>JIU</td>
<td>$DC_{DE}$</td>
<td>0.0228</td>
</tr>
<tr>
<td></td>
<td>$DC_{IE}$</td>
<td>0.0267</td>
</tr>
<tr>
<td></td>
<td>$DC_D$</td>
<td>0.0186</td>
</tr>
<tr>
<td></td>
<td>$DC_I$</td>
<td>0.0221</td>
</tr>
<tr>
<td>Moneyjar</td>
<td>$DC_{DE}$</td>
<td>0.0226</td>
</tr>
<tr>
<td></td>
<td>$DC_{IE}$</td>
<td>0.0237</td>
</tr>
<tr>
<td></td>
<td>$DC_D$</td>
<td>0.032</td>
</tr>
<tr>
<td></td>
<td>$DC_I$</td>
<td>0.0331</td>
</tr>
</tbody>
</table>

Table 5 : $R^2$ obtained with the log of a coupling value

Spearman Correlation study (rank statistic)

Further, we calculated the correlation degree (according to Spearman) between the cohesion metrics and coupling in the chosen systems. Table 6 gives the results that we have got.

The aim of this research was to identify a correlation (negative) between the cohesion metrics and coupling metric we have chosen. This tests main goal is to find out if the connectedness is significantly lower than 0 (in the statistical sense) for a negative dependence. A statistical research was conducted. The statistical research must then be correlated to a Student variable computed with $n-2$ freedom degrees, and where $n$ is the size of the example. The $P$-value shows the probability of getting such a value under the null hypothesis of absence of dependence. In general, if $P$-value $< 0.05$ (error margin), we come to a conclusion that a negative dependence is significant. Hence, for the group of tested systems and from the values of table 6, only the moneyjar system has $P$-values $> 0.05$ for all combinations (cohesion metric – coupling metric). We examine values of 0.48996, 0.46740, 0.4649, and 0.442451 for, respectively, cohesion metrics DCIE, DCDE, DCI, DCD correlated to the coupling metric CBO.
For the remaining chosen systems, the P-values are all < 0.05 for the whole group of combinations (cohesion metric – coupling metric). As per table 6, all systems show a significant negative dependence between cohesion and coupling. But this is not possible with moneyjar system. The reason behind this exception is that this particular system has got less number of classes [20] than the other systems. Hence we can come to a conclusion that to observe significant negative dependency is better to select systems with high number of classes. From the above results, it can be said that there is a correlation between cohesion metric and coupling metric.

Other than this it is also observed that if the number of classes are high in a system then the dependency level between cohesion and coupling (Non linear dependency relations) can be confirmed easily. Different kind of systems were selected to prove the correlation between cohesion and coupling and is proved. But there are many other kind of systems on which it is not tested. Hence it is better to prove the same on other systems also before we give any global declaration.

### Table 6: Results of the Spearman rank statistic method

<table>
<thead>
<tr>
<th>System</th>
<th>Statistic</th>
<th>DCIE-CBO</th>
<th>DCIE-CBO</th>
<th>DCI-CBO</th>
<th>DCI-CBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>S.Coeff</td>
<td>0.354545</td>
<td>0.354545</td>
<td>0.354545</td>
<td>0.354545</td>
</tr>
<tr>
<td>...</td>
<td>Test stat</td>
<td>-2.786937</td>
<td>-2.8258</td>
<td>-2.786937</td>
<td>-2.8258</td>
</tr>
<tr>
<td>...</td>
<td>P-value</td>
<td>0.003659</td>
<td>0.003299</td>
<td>0.003659</td>
<td>0.003299</td>
</tr>
</tbody>
</table>

**XI. Conclusion**

With the help of this research we introduced a new criterion and gave a better, revised definition of class cohesion [5, 6]. Common objects parameters are the new criteria which is introduced and also validated in this paper. This becomes the new way to measure class cohesion. We enhanced a cohesion measurement tool for Java programs to automate the calculation of the class cohesion metrics that we propose. Different kinds of systems were taken for the experiment to prove that the new criterion and the proposed metrics for class cohesion give the best assessment. These systems are very much different in size and domain. In this test many number of classes were analyzed. After all the experiments it was understood that the extended metrics with the help of new criterion is capable of finding out more pairs of connected methods. In addition to the above experiment one more was also conducted. It helped to validate our new metrics. This was helpful to prove the correlation between cohesion and coupling. To the second step we got a chance to observe several hundreds of classes. From this second test it was found in the selected systems that there exists a negative correlation between cohesion and coupling. More over through the results we could see that if the number of classes in a system is high then the dependency relation between cohesion and coupling can be confirmed easily.

**References**