CHAPTER 4

HIGH PRECISION COHESION METRIC

4.1 INTRODUCTION

High quality software design, among many other principles, should obey the principle of high cohesion. In structural development techniques, cohesion is defined as the measure of the degree to which the elements of a module that belong together (Briand et al 1998b). A module that is highly cohesive performs one basic function and cannot be split into separate modules easily. Highly cohesive modules are understandable, modifiable and maintainable (Al Dallal & Briand 2012).

In OO paradigm, class cohesion can be defined as the relatedness of its members – methods and attributes. Researchers have introduced several metrics to measure the class cohesion. The cohesion metrics are based on different perspectives. Chidamber & Kemerer (1991) proposed the first cohesion metric. They presented an inverse metric, thus measuring the lack of cohesion (LCOM1). This metric measures the number of pairs of methods that do not share any common attribute. Chidamber and Kemerer revised their metric and introduced LCOM2 (1994), which computes the difference between the number of pairs of methods not sharing any common attribute and those sharing at least one attribute. Li & Henry (1993) proposed their version of lack of cohesion, LCOM3 as the number of connected components in a graph with methods as nodes and edges between each pair of methods with at least one common attribute. Hitz & Montazeri (1996) proposed
LCOM4, as an enhancement of LCOM3, which includes edges for method invocations. Henderson-Sellers (1996) introduced LCOM5 which included the number of distinct attributes accessed by each method.

Connectivity (Co), a direct cohesion metric was introduced by Hitz & Montazeri (1995). Briand et al (1998b) proposed Coh which relies on the number of attributes used by each method. Bieman & Kang (1995) introduced TCC (Tight class cohesion) and LCC (Loose class cohesion) which captured the methods connected through attributes directly and indirectly. Badri (2004) proposed DC₀ and DC₁ which are variations of TCC and LCC. DC₀ includes direct method invocations. DC₁ includes both direct and indirect method invocations.

Metrics based on the similarity between methods have been proposed. Bonja & Kidanmariam (2006) defined Class Cohesion (CC). This metric was enhanced by Fernandez & Pena (2006) in Sensitive Class Cohesion Metric (SCOM). Bansiya et al. (1999) defined Cohesion among Methods in a class (CAMC) as a modified version of Co. Counsell et al in (2006) introduced Normalized Hamming Distance (NHD) and Scaled Normalized Hamming Distance (SNHD) based on how many attributes each method accesses. Al Dallal & Briand (2012) proposed Low-level design Similarity based Class Cohesion (LSCC) which computes cohesion based on the number of methods that access each attribute.

Several studies have been undertaken to validate the cohesion metrics. Hitz & Montazeri (1996) illustrated that Lack Of Cohesion metric LCOM1 has many anomalies. Etzkorn et al (1999) confirm that LCOM2 has ambiguous definition. Chae et al (2000) report that LCOM metrics, TCC and LCC do not compute cohesion metrics intuitively. This is because these metrics include constructors, destructors and accessors in the computation of
cohesion. Zhou & Xu (2008) observe that CBMC also gives inconsistent results in certain cases.

4.2 LIMITATIONS IN EXISTING COHESION METRICS

LCOM1 measures the number of pairs of methods that do not share any common attribute. LCOM2 is computed as the number of pairs of methods that do not share any common attribute minus the number of method pairs that share at least one attribute. For all method pairs I_i and I_j in a class, LCOM1 is given by Equation (4.1) and LCOM2 is given by Equation (4.2).

\[
LCOM1 = \{ |I_i, I_j|, I_i \cap I_j = \emptyset \} \tag{4.1}
\]

\[
P = \{ |I_i, I_j|, I_i \cap I_j = \emptyset \}
Q = \{ |I_i, I_j|, I_i \cap I_j \neq \emptyset \}
LCOM2 = \begin{cases} 
|P| - |Q|, & \text{if } |P| > |Q| \\
0, & \text{otherwise}
\end{cases} \tag{4.2}
\]

LCOM1 and LCOM2 have certain limitations. They lack a normalized form, as the values are directly proportional to the number of methods in a class. This makes it very difficult to compare LCOM1 and LCOM2 values between a pair of classes. Consider the two classes in Figure 4.1. Each class method uses exactly one attribute of their respective class. Though it seems that both classes have relatively similar cohesion, the values are skewed due to the number of methods in the two classes. Here, \( LCOM1(\text{class1}) = LCOM2(\text{class1}) = 1, LCOM1(\text{class2}) = LCOM2(\text{class2}) = 6. \)
To compute LCOM3 a graph is constructed with methods as nodes and edges between methods sharing at least one attribute. LCOM3 is given as the number of connected components in the graph. LCOM4 is also computed similarly except that during the construction of the graph, edges are added for method invocations also. Since LCOM3 and LCOM4 count the connected components the range of their values is lesser compared to LCOM1 and LCOM2. However the cohesion values are still not precise. For the example shown in Figure 4.1, $LCOM3(class1) = LCOM4(class1) = 2$ and $LCOM3(class2) = LCOM4(class2) = 4$. The values obtained for LCOM3 and LCOM4 for class1 and class2 are different, though the classes have similar method-attribute interaction. A better example to highlight the drawback of LCOM3 and LCOM4 is given in Figure 4.2. Class 3 contains 7 attributes and 6 methods. Each method accesses a unique attribute along with one common attribute a4. There is very less cohesion between the methods. The common attribute could be a debug variable or a static counter variable. For this example, $LCOM3(class3) = LCOM4(class3) = 1$. The LCOM3 and LCOM4 values indicate that the class has the highest possible cohesion which is not the case.
TCC and LCC provide another way to measure the cohesion of a class by analyzing the relations between methods. Two methods ‘a’ and ‘b’ are related if they both access the same class level variable or if the call trees starting at ‘a’ and ‘b’ access the same class level variable. Once the graph is drawn with edges depicting relations, two methods are said to be directly connected if they have an edge between them. If they are connected through other methods then they are indirectly connected. If there are N methods, NDC represents number of direct connections; NIC represents the number of indirect connections, then TCC is given in Equation (4.3) and LCC is given in Equation (4.4).

\[
TCC = \frac{NDC}{N \times (N - 1)}
\]  
(4.3)

\[
LCC = \frac{(NDC + NIC)}{N \times (N - 1)}
\]  
(4.4)

DC_D and DC_1 are similar to TCC and LCC. They consider an additional relationship of method invocation. Methods ‘a’ and ‘b’ are said to be connected if the two methods directly or indirectly invoke a third method. If NTC represents the number of transitive connections, then DC_D is given by Equation (4.5) and DC_1 is given by Equation (4.6).
\[ D_{DC} = \frac{(NDC + NTG)}{(N \times (N - 1))} \]  \hspace{1cm} (4.5)

\[ D_{CI} = \frac{(NDC + NIC + NTG)}{(N \times (N - 1))} \]  \hspace{1cm} (4.6)

The value for TCC, LCC, DC_D and DC_I is 1 for the example in Figure 4.2. The reason behind the same value is that all these metrics provide highest cohesion between the sharing methods of the class even if just one attribute is shared across them.

CC and SCOM consider the level of similarity between the pair of methods. Similarity between methods is calculated using the number of common attributes used between the pairs of methods. CC and SCOM differ in the denominator used to divide the common attributes. CC uses the distinct attributes between the methods whereas SCOM uses the minimum number of attributes used in the two methods. If \( N \) is the number of methods in a class, \( I_i \) and \( I_j \) refer to the method pairs, CC is given by Equation (4.7) and SCOM is given by Equation (4.8).

\[ CC = \frac{\sum_{i,j} |I_i \cap I_j|}{(N \times (N - 1))} \]  \hspace{1cm} (4.7)

\[ SCOM = \frac{\sum_{i,j} |I_i \cap I_j|}{\text{Min}(|I_i|, |I_j|)} \]  \hspace{1cm} (4.8)

CC and SCOM have certain limitations. When there are many method pairs that access very few attributes of the class, but have sufficient commonality, they tend to bloat the cohesion value. In class 4 shown in Figure 4.3, the class has 5 attributes and 5 methods. Attribute a1 is accessed by methods m1, m2, m3 and m4, while the other attributes are accessed by
method m5 alone. On careful observation, it can be confirmed that the class has poor cohesion, since majority of the attributes are not shared across the methods. But, both CC and SCOM give better than average value. Here $CC(\text{class}4) = SCOM(\text{class}4) = 0.6$ (Max cohesion possible is 1.0). So the cohesion property is not completely captured by the metrics.

![Class 4](image)

**Figure 4.3 Example for SCOM and CC**

All above reasons contributed to the motivation behind the need to explore a better cohesion metric.

### 4.3 PROPOSAL OF HPCM

HPCM is a direct cohesion metric. It captures the cohesion of a class using method-attribute interaction. The extent of the method-attribute interactions are identified using Average Attribute Usage and Link Strength. The notations followed for proposing HPCM is derived from Briand et al.’s Unified Framework for Cohesion (UFC) (1998b).

#### 4.3.1 Average Attribute Usage

The Average Attribute Usage (AAU) computes the average number of attributes used by each method of the class. Only public, non-inherited methods of a class are considered. As per UFC, the total of such methods are arrived at using the following approach. Let ’c’ be the class in consideration.
Then $M_f(c)$ is the set of non-inherited, overriding or newly implemented methods of $c$. Further $M_{pub}(c)$ is the set of public methods of $c$. Public non-inherited or overridden methods are given by $M_f(c) \cap M_{pub}(c)$.

The total number of methods is the cardinality of such a set. It is given by $|M_f(c) \cap M_{pub}(c)|$. The attributes referenced by a method is given by $AR(m)$, where $m$ is the method. Hence total attributes referenced is given by $\sum AR(m)$, for each ‘m’ in the set $M_f(c) \cap M_{pub}(c)$. Hence the AAU is given as Equation (4.9).

$$AAU(c) = \frac{\sum AR(m)}{|M_f(c) \cap M_{pub}(c)|}$$  \hspace{1cm} (4.9)

The AAU for class 4 in Figure 4.3 is 1.6.

4.3.2 Link Strength

Link Strength is a concept to capture the level of interaction between a pair of methods based on the number of attributes commonly used between them. The Link Strength (LS) is based on the AAU. Link Strength for a pair of methods m1 and m2 is given by Equation (4.10).

$$LS_{m1m2} = \begin{cases} \frac{|AR(m1) \cap AR(m2)|}{AAU}, & \text{if } |AR(m1) \cap AR(m2)| \leq AAU \\ 1, & \text{if } |AR(m1) \cap AR(m2)| > AAU \end{cases}$$  \hspace{1cm} (4.10)

The Link Strength concept of HPCM is explained using the LCOM4 connectivity graph. LCOM4 considers the methods as nodes of a graph. Two nodes are connected with an edge if the underlying methods have at least one attribute in common. Consider the example of class 5 in Figure 4.4. The class
consists of 6 attributes and 3 methods. A LCOM4 graph is drawn to depict the same in Figure 4.5.

![Class 5 diagram](image)

**Figure 4.4 Example for Link Strength**

![LCOM4 graph](image)

**Figure 4.5 LCOM4 graph**

As expected the graph shows a single connected component giving an LCOM4 value of 1. On observing the class it can be seen that method m1 and method m2 share a single attribute a2; method m2 and method m3 share two attributes a3 and a4. Method pairs m1 and m3 share three attributes a1, a5, a6. So the link between m1 and m2 is not the same as that of m2 and m3, which in turn is different from the link between m1 and m3. The Link Strength metric is designed to capture this difference in the strength of the links between methods. The graph for LCOM4 metric has been modified to include link strength in Figure 4.6. It is clear that though all three methods are linked, the level or strength of their links are different and this is captured in LS.
4.3.3 Definition of HPCM

The new cohesion metric has been named High Precision Cohesion Metric (HPCM) to denote its capability to distinguish and assign precise cohesion values for a wide variety of classes with various method-attribute interactions. It depends on the earlier proposed concepts - AAU and LS. The formula for HPCM is given in Equation (4.11).

\[
HPCM(c) = \frac{2 \times \sum LS_{m_im_j}}{|M_f(c) \cap M_{pub}(c)| \times (|M_f(c) \cap M_{pub}(c)| - 1)}
\]  

(4.11)

HPCM is an average of LS of all method pairs in the class. To compute this average, the total of all LS in the numerator is computed which is given as \(LS_{m_im_j}\). Here all public, non inherited, overridden and newly implemented methods are considered. The denominator denotes the total available method pairs and is given by \(n \times (n - 1)/2\) where ‘n’ is the total number of methods.

The total number of methods is given by \(M_f(c) \cap M_{pub}(c)\) and hence results in the denominator of 
\(|M_f(c) \cap M_{pub}(c)| \times (|M_f(c) \cap M_{pub}(c)| - 1)\). The final division by 2 gets back to the numerator. Below is an example showing the calculation of HPCM for the class 5 given in...
Figure 4.4. The first step would be to decide how many methods are to be considered for the metric. It can be seen that $M_f(c) \cap M_{pub}(c) = 3$. The next step would be to find AAU.

The first method uses 4 attributes, the second uses 3 attributes and the third uses 5. Thus the total attribute usage is 12 and the average usage would be $(4 + 3 + 5)/3 = 4$. Hence AAU = 4. The link strength between each pair of methods is computed next. Since the number of methods considered is 3, it can be seen that there are $3 \times (3 - 1)/2 = 3$ method pairs. The calculation steps for the method pair m1 and m2 are shown here. Table 4.1 shows the link strength values for all method pairs for class 5 shown in Figure 4.4.

$$LS_{m_1m_2} = \frac{|AR(m_1) \cap AR(m_2)|}{AAU} = \frac{1}{4} = 0.25$$

<table>
<thead>
<tr>
<th>Method Pairs</th>
<th>Common Attributes</th>
<th>Link Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1m_2$</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>$m_1m_3$</td>
<td>3</td>
<td>0.75</td>
</tr>
<tr>
<td>$m_2m_3$</td>
<td>2</td>
<td>0.50</td>
</tr>
<tr>
<td>Total</td>
<td>6.00</td>
<td>1.50</td>
</tr>
</tbody>
</table>

The High Precision Cohesion Metric is computed as given below

$$HPCM(c) = \frac{2 \times \sum LS_{m_mm_i}}{|M_f(c) \cap M_{pub}(c)| - 1} = \frac{2 \times 1.50}{3 \times (3 - 1)} = \frac{1.50}{3} = 0.50$$
4.4 CRITERIA OF APPLICATION

Metrics tend to be plagued by poor definition, assumptions and lack of application criteria. This causes difficulty in using the metric and can result in multiple interpretations by different users. In this section the criteria of application for HPCM are listed.

Class Level Design Metric

Metrics can be specified at various levels like system, class or method level. HPCM is a class level metric. Metrics can also be classified based on the phase in which they are computed in the life cycle of the software product. HPCM is a design phase metric.

Inheritance

Public methods which are directly implemented in the class are considered. This also includes methods that are inherited and overridden. Methods that are inherited but not overridden and those that are private are not considered. Attributes that are introduced by the class are considered. The attributes that are inherited are not considered for computation of HPCM.

Methods not Accessing any Attribute

Two kinds of methods are not considered for the computation of HPCM, as they do not access any class attribute. Static members accessing only static attributes are not allowed to access any class attributes. Similarly pure virtual methods not implemented in this class will not access any attribute.
Attributes not Accessed by any Method

Attributes defined in a class but not used by any of its methods are not considered for the computation of HPCM. Such attributes are used in inheritance where the child class might use an attribute not used by its parent. However such attributes should not be considered while computing the cohesion of the class.

Access Methods

Access methods are used to get and set attributes of a class. The get/set methods typically deal with a single attribute. They are not considered for computing HPCM as it would result in an artificial decrease of cohesion value.

Constructor and Destructor

Constructor and destructor methods access many attributes of the class. The methods are designed to initialize the attributes of the class when the object of the class is created and to destroy its members when the object is destroyed. They are not considered in the computation of HPCM as it would result in artificial increase of cohesion value.

Method Invocations

Method invocations are not considered as contributing to the relationship between them. If method M_i calls M_j it does not add to their link strength. However, the transitive access of attributes due to method invocations is considered for link strength calculation. If method M_i calls method M_j and if M_j accesses an attribute a_k, then it is considered that AAU and Link Strength.
4.5 MATHEMATICAL VALIDATION OF HPCM

The cohesion properties introduced by Briand et al. (1998b) in his Unified Framework for Cohesion (UFC) are evaluated for HPCM. The four properties characterize cohesion in a reasonably intuitive and rigorous manner. These properties may not be sufficient to ensure that a measure which fulfills them all will be a useful metric. However, it is likely that a measure which does not fulfill them is probably not defined well metric.

Non-Negativity and Normalization

The minimum value of HPCM is 0. This is a result of a class with no attributes or when there are no common attributes between all pairs of methods. The maximum value of HPCM is 1, when all methods share at least AAU attributes between them. Thus $HP_{CM}(c) \in [0,1]$ where ‘c’ is the class of an object oriented system C.

Null Value and Maximum Value

Let c be the class of an object oriented system C and IR is the set of all interactions in the class. The cohesion of a class is null if IR is empty, that is there are no shared attributes between the methods. Consider class 6 in Figure 4.7. It can be seen that IR results in an empty set or link strength is 0 for all the method pairs and hence $HP_{CM}(class6) = 0$, the minimum value of HPCM. Alternatively the class 7 in Figure 4.7 results in a set with all possible members in the set IR. This gives $HP_{CM}(class7) = 1$, the maximum value of HPCM.
Figure 4.7 Example for null and maximum value of HPCM

Monotonicity

Consider a class $c$ in the object oriented system $C$. If the class is modified to form a new class $c'$ which is identical to $c$ except that few more attributes are shared by methods. When new relationships are added to the existing method-method relationships the link strength becomes higher with increase in numerator. Hence HPCM also increases. $HPCM(c) \leq HPCM(c')$. So HPCM satisfies the monotonicity property.

Merging Unconnected Classes

Let $C$ be an object oriented system and $c_1, c_2 \in C$. Let $c'$ be the class which is the union of $c_1$ and $c_2$. Let $C'$ be the object-oriented system which is identical to $C$ except that $c_1$ and $c_2$ are replaced by $c'$. If no relationships exist between classes $c_1$ and $c_2$ in $C$, then

$max(HPCM(c_1), HPCM(c_2)) \geq HPCM(c')$.

4.6 FAULT PREDICTION USING COHESION METRICS

Cohesion metrics have been experimented for fault prediction similar to other metrics. Two types of experiments are performed. The first
experiment evaluates whether HPCM is contributing new information in computing the cohesion of a class. The second experiment is focused on building models to validate HPCM’s ability to predict faults compared to other cohesion metrics.

4.6.1 Dataset

Four open source projects written in Java are considered from the sonar source repository website (http://nemo.sonarsource.org). The fault count is also available in the repository. The faults are classified as blocker, critical, major, minor and information. For this research work, blocker, critical and major faults are considered. The projects chosen are Spojo, Maven Plugin, XDoclet, and Sonar Plugin. Each project had approximately 20 classes. The following cohesion metrics are taken for this empirical validation - LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, TCC, LCC, CC, SCOM, and the proposed HPCM. The Table 4.2 shows the descriptive statistics of the metric values chosen for the study indicating the minimum, maximum, mean, median, standard deviation, 25th Quartile value and 75th Quartile values for the metrics. LCOM1 and LCOM2 have the broadest range and SD for metric values. Metric values are normalized for TCC, LCC, SCOM, HPCM and CC. The difference between the mean and median values gives insight on the nature of the data distribution.

The Automated Metrics Tool (AMT) described in the previous chapter is used to compute the cohesion metrics. HPCM is added to the list of metrics computed by the tool. The source code of the projects are downloaded and used to extract the metrics.
Table 4.2 Descriptive Statistics of the Metric Values

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Min</th>
<th>25Q</th>
<th>Mean</th>
<th>Median</th>
<th>75Q</th>
<th>Max</th>
<th>StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCOM1</td>
<td>0.00</td>
<td>3.00</td>
<td>72.94</td>
<td>13.00</td>
<td>63.00</td>
<td>683.00</td>
<td>139.02</td>
</tr>
<tr>
<td>LCOM2</td>
<td>0.00</td>
<td>1.00</td>
<td>62.64</td>
<td>7.00</td>
<td>44.00</td>
<td>663.00</td>
<td>133.33</td>
</tr>
<tr>
<td>LCOM3</td>
<td>1.00</td>
<td>2.00</td>
<td>4.06</td>
<td>3.00</td>
<td>4.00</td>
<td>19.00</td>
<td>3.80</td>
</tr>
<tr>
<td>LCOM4</td>
<td>1.00</td>
<td>1.00</td>
<td>2.88</td>
<td>2.00</td>
<td>3.00</td>
<td>17.00</td>
<td>2.94</td>
</tr>
<tr>
<td>LCOM5</td>
<td>0.00</td>
<td>0.63</td>
<td>0.75</td>
<td>0.79</td>
<td>0.91</td>
<td>2.00</td>
<td>0.33</td>
</tr>
<tr>
<td>TCC</td>
<td>0.00</td>
<td>0.12</td>
<td>0.29</td>
<td>0.17</td>
<td>0.33</td>
<td>1.00</td>
<td>0.27</td>
</tr>
<tr>
<td>LCC</td>
<td>0.00</td>
<td>0.13</td>
<td>0.35</td>
<td>0.24</td>
<td>0.50</td>
<td>1.00</td>
<td>0.33</td>
</tr>
<tr>
<td>SCOM</td>
<td>0.00</td>
<td>0.12</td>
<td>0.29</td>
<td>0.19</td>
<td>0.40</td>
<td>1.00</td>
<td>0.27</td>
</tr>
<tr>
<td>CC</td>
<td>0.00</td>
<td>0.06</td>
<td>0.20</td>
<td>0.13</td>
<td>0.31</td>
<td>1.00</td>
<td>0.22</td>
</tr>
<tr>
<td>HPCM</td>
<td>0.00</td>
<td>0.02</td>
<td>0.20</td>
<td>0.11</td>
<td>0.25</td>
<td>1.00</td>
<td>0.26</td>
</tr>
</tbody>
</table>

4.6.2 Results and Discussion

A correlation study is performed to understand the relationship between the cohesion metrics considered. Pearson coefficient is computed for each pair of cohesion metrics. The required assumptions for using Pearson coefficient are met – the data couples correspond to the same object and are normally distributed. It gives the correlation between two metrics in the range of +1, indicating high positive correlation to -1, indicating high negative correlation and 0 indicating no correlation. Table 4.3 shows the Pearson coefficient for the cohesion metrics which have significant p-value. Metrics capturing the same dimension of cohesion will have high correlation between them. LCOM1, LCOM2, LCOM3, and LCOM4 have correlation coefficient in excess of 0.80. Significant correlation also exists between LCOM5, TCC, LCC, SCOM and CC. High correlation coefficient which also have significant p-value (p<0.0001) have been highlighted.
Table 4.3 Pearson Coefficient for Cohesion Metrics

<table>
<thead>
<tr>
<th>Metrics</th>
<th>LCOM2</th>
<th>LCOM3</th>
<th>LCOM4</th>
<th>LCOM5</th>
<th>TCC</th>
<th>LCC</th>
<th>SCOM</th>
<th>CC</th>
<th>HPCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCOM1</td>
<td>0.99</td>
<td>0.91</td>
<td>0.81</td>
<td>0.24</td>
<td>-0.33</td>
<td>-0.33</td>
<td>-0.36</td>
<td>-0.31</td>
<td>-0.14</td>
</tr>
<tr>
<td>LCOM2</td>
<td>1.00</td>
<td>0.92</td>
<td>0.82</td>
<td>0.24</td>
<td>-0.33</td>
<td>-0.33</td>
<td>-0.36</td>
<td>-0.31</td>
<td>-0.16</td>
</tr>
<tr>
<td>LCOM3</td>
<td>1.00</td>
<td>0.82</td>
<td>0.30</td>
<td>-0.45</td>
<td>-0.46</td>
<td>-0.49</td>
<td>-0.43</td>
<td>-0.27</td>
<td></td>
</tr>
<tr>
<td>LCOM4</td>
<td>1.00</td>
<td>0.25</td>
<td>-0.39</td>
<td>-0.40</td>
<td>-0.41</td>
<td>-0.35</td>
<td>-0.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCOM5</td>
<td>1.00</td>
<td>-0.64</td>
<td>-0.58</td>
<td>-0.71</td>
<td>-0.75</td>
<td>-0.53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCC</td>
<td>1.00</td>
<td></td>
<td>0.96</td>
<td>0.95</td>
<td>0.86</td>
<td>0.55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCC</td>
<td>1.00</td>
<td></td>
<td>0.91</td>
<td>0.74</td>
<td></td>
<td>0.41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOM</td>
<td>1.00</td>
<td></td>
<td>0.91</td>
<td></td>
<td>0.53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td>1.00</td>
<td></td>
<td></td>
<td>0.31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPCM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If a metric has low Pearson coefficient with other metrics, it indicates that the metric is capturing the cohesion property in a unique way compared to other cohesion metrics. It can be seen that HPCM has low to average correlation coefficient with other cohesion metrics. HPCM being a direct measure of cohesion has a negative correlation with the inverse cohesion metrics like LCOM1. In the second analysis, the fault prediction effectiveness of each cohesion metric is measured. Simple linear regression explained in Chapter 2 has been used for this study. The preconditions for applying simple linear regression are met. MedCalc tool (http://www.medcalc.org/index.php) is used for the study. The results of the simple linear regression are given in the Table 4.4. HPCM is a significant contributor with the RMSE of 0.37.
The second set of experiments have been conducted by developing two fault prediction models, one with all cohesion metrics except HPCM and another including HPCM. The goal of this analysis is to confirm that when HPCM is included, the fault prediction accuracy improves in models developed using other cohesion metrics. Three different prediction techniques – Multiple linear regression, Bayesian network and Decision trees are chosen. The assumptions for using these prediction techniques are met. Multiple linear regression is described in Chapter 2. Backward stepwise elimination technique has been used. Table 4.5 shows the details of the prediction with and without HPCM for multiple linear regression. It can be seen that by adding HPCM the root mean square error improves from 3.85 to 2.90.

**Table 4.5 Results of Multiple Linear Regression**

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without HPCM</td>
</tr>
<tr>
<td>Multiple Linear Regression</td>
<td>3.85</td>
</tr>
</tbody>
</table>
Bayesian Network is a probabilistic graphical model that represents a set of random variable and their conditional dependencies. It has been used to map the cohesion metrics and fault count. On providing the cohesion metrics the model will output the probability of occurrence of faults. Decision Trees represent decision making with branches relating to decisions and leaves representing the result. Table 4.6 shows the details of the prediction with and without HPCM for Decision tree and Bayesian network. Weka tool (http://www.cs.waikato.ac.nz/ml/weka) is used to perform the analysis. The prediction accuracy of models with HPCM is better compared to the prediction accuracy of models without HPCM.

**Table 4.6 Results of Bayesian and Decision Tree Models**

<table>
<thead>
<tr>
<th>Models</th>
<th>Prediction Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without HPCM</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>71</td>
</tr>
<tr>
<td>Bayesian Net</td>
<td>50</td>
</tr>
</tbody>
</table>

4.7 **SUMMARY**

The limitations of existing cohesion metrics are identified. A new cohesion metric, the High Precision Cohesion Metric (HPCM) is proposed to address the short comings of the earlier cohesion metrics. The Unified Framework for Cohesion is used to present the proposed metric. The criteria for applying the metric are described. A mathematical validation is carried out to evaluate HPCM on the properties proposed by Briand et al. Empirical validation of HPCM is performed with other cohesion metrics, using open source Java projects. The results indicate that HPCM performs better, both as a standalone metric and as a contributing metric in a metric suite.