CHAPTER 6
Assessment of the Class Testability

6.1 Introduction

Software testability is an external quality attribute of the software that provides a guideline for testing. It is affected by a number of factors like the representation of requirements, the required validity, the tools used, and the process [32]. Assessment of software testability may help in planning the testing activities and making the testing process more effective. In this chapter, some of the factors of testability are identified for object oriented software. The design metrics that are related to testability are identified and evaluated with the help of four case studies of open source java systems, for which JUnit test cases exist. An Eclipse plugin was developed to extract the value of the design metrics and the test metrics from the source systems. The test metrics were extracted from the JUnit test classes and their correlation was found with the design metrics. The correlation between the design metrics and the test metrics shows that most of the metrics are significantly correlated to the test metrics. The test metrics help in assessing the testing effort, and through the testing effort an assessment of testability can be made.

The motivation for this study is: 1) there are many theories on software testability but the empirical studies are few. 2) Software testability is one attribute of software quality. Hence, assessing software testability will help in improving the software quality. 3) Assessing software testability helps in planning the testing activities. It also provides the information about the testing effort required to test a system [108]. 4) JUnit [14] has become popular as a part of the agile software
development [39], test driven development [16], and extreme programming [15] methodologies. A number of empirical studies [65, 121, and 150] have been published to predict and assess various attributes of software quality, such as maintainability, fault proneness, and so on. These studies use different statistical methods or neural networks to predict the software quality attributes. Software testability also is a software quality attribute that is a part of a software maintainability attribute. The similarity between these studies and our study is that they also use design metrics and the statistical method to assess the software testability.

6.2 Software Testability

Although, there are a number of definitions of software testability but we abide by the definition of software testability given by ISO [77]. The attributes of software that we consider are the design metrics of the object oriented software. This study investigates the effect of design metrics on the testing effort. The design metrics are divided into two categories, test case generation factors and test case construction factors, as defined by Bruntink and Deursen [32].

6.2.1 Test Case Generation Factor

Bruntink and Deursen [32] define the test case generation factor as the factor that influences the number of test cases required to test a system. The testing criterion and the design metrics determine how many test cases are created. Object oriented software has different features such as class, inheritance, polymorphism, dynamic binding, and so on. These features have a bearing on the testing effort and, hence,
on the software testability. The features can be classified as the test case generation factors or test case construction factors, which are discussed below.

1) Testing a class involves traditional testing and state-based testing. State testing of a class requires testing all the states of an object of the class according to the state testing criterion. Hence, the number of discrete states of a class is a test case generation factor.

2) Inheritance is a mechanism that allows a class to be derived from an already existing class. Inheritance has a bearing on the testing effort, as it will increase the number of test cases because the child class tests both the inherited methods and its own methods. Hence, inheritance is a test case generation factor.

3) Polymorphism is a feature of object oriented languages through which an object of a class can belong to more than one class. Consider two classes, A and B, where A is the parent class of B. In this case the objects of class B are also the objects of class A. Since polymorphism increases the number of test cases, it is a test case generation factor. However, when polymorphism is applied in the case where class A and class B are independent of one another, then the polymorphic methods are treated like any other method in the assessment.

6.2.2 Test Case Construction Factor

The test case construction factor is defined by Bruntink and Deursen [32] as the factor that influences the effort needed to construct each test case. Constructing a test case sometimes can be difficult, such as: 1) in state testing of a class the object goes into a dead state or some states are not reachable and many other cases of this type; 2) class testing needs initialization of methods of another class if a class uses methods of another class; and 3) in the case of exception handling, and so on.
The features of object oriented software can be divided into the aforementioned two categories, that is, either test case generation factors or test case construction factors. All the design metrics that are considered for this study belong to some of the object oriented features.

6.3 JUnit Testing Framework

JUnit is an open source framework which has been designed for the purpose of writing and running tests in Java programming language. It was originally written by Erich Gamma and Kent Beck [14]. JUnit has gained a lot of popularity as a part of test driven development [16], agile software development [39], and extreme programming methodology [15]. JUnit defines how to write the test cases and provides the tool to run them. JUnit helps in testing a java class by writing the corresponding JUnit class. A number of other frameworks exist for class testing in other languages like CPPUnit (for C++) etc. An example java class (calculator.java) and its JUnit test class (calculatorTest.java) is shown in Figures 6.1 and 6.2 respectively.

To test our “calculator.java” class we define “calculatorTest.java” as a subclass of TestCase. The JUnit test class contains various test methods to test the methods of the source class. A JUnit test method does not contain any parameters. In writing the test cases in a JUnit class a number of asserts are used. For example assertEquals (expected output, actual output), assertEquals(message, expected output, actual output), etc. The calculatorTest class in figure 6.2 uses assertEquals() to compare the expected output and the actual output. If they are equal the test case passes otherwise the test case fails. In Java, classes are contained in packages. Hence, the source class and the test class can be kept in
same or different packages. In this example, calculator is the source class and calculatorTest is the test class. There can be a number of source classes and a number of test classes corresponding to the source classes in a given system. In the source system used for this study, the source classes are kept in the source package and the test classes are kept in the test packages.

```java
public class calculator {
    public calculator()
    {
    }
    public int add(int a, int b)
    {
        return a+b;
    }
    public int subtract(int a, int b)
    {
        return a-b;
    }
}
```

```java
import junit.framework.TestCase;
public class calculatorTest extends TestCase
{
    private calculator cal = new calculator();
    private int a = 4;
    private int b = 3;
    public void testAdd() {
        assertEquals(7, cal.add(a,b));
    }
    public void testSubtract() {
        assertEquals(1, cal.subtract(a,b));
    }
}
```

**Figure 6.1: calculator.java class**

**Figure 6.2: calculatorTest.java class**

### 6.4 Experimental Design

In this section, we evaluate whether the OO design metrics can be used to assess the testability of a class. We identify the OO design metrics under consideration and formulate a hypothesis to be analyzed. Subsequently, the test metrics based on JUnit test classes are analyzed. This section also describes the statistical method used to evaluate the hypothesis.

#### 6.4.1 Object Oriented Metrics

Object oriented metrics chosen for this analysis can be divided into five categories: size, coupling, cohesion, inheritance, and polymorphism. These metrics are defined below and are listed in Table 6.1.
Size metrics

The size metrics measure the size of the system in terms of lines of code, attributes and methods included in the class. As these metrics capture the complexity of the class hence they can give an insight into the testability of the class.

**Lines of code per class (LOC):** It counts the total number of lines of code (non-blank and non-comment lines) in the class [63].

**Number of Attributes per Class (NOA):** It counts the total number of attributes defined in a class [63].

**Number of Methods per Class (NOM):** It counts the number of methods defined in a class [63].

**Weighted Methods per Class (WMC):** The WMC is the count of the sum of the McCabe’s Cyclomatic Complexity for all the methods in the class. If method complexity is one for all the methods, then WMC = n, the number of methods in the class [35].

Cohesion Metrics

Cohesion measures the degree to which the elements of a module are functionally related. A strongly cohesive module does little or no interaction with other modules and implements the functionality which is related to only one feature of the software.

This study considers the following three cohesion metrics.

**Lack of Cohesion in Methods (LCOM):** Lack of Cohesion (LCOM) [35] measures the cohesiveness of the class. It is defined as below: Let M be the set of methods and A be the set of attributes defined in the class. $M_a$ is the number of methods that access a. Mean be the mean of $M_a$ over A. Then,
**LCOM** = (Mean -|M|)/(1 - |M|)

**Information flow based Cohesion (ICH):** ICH for a class is defined as the number of invocations of other methods of the same class, weighted by the number of parameters of the invoked method [93].

**Tight Class Cohesion (TCC):** The measure TCC is defined as the percentage of pairs of public methods of the class with common attribute usage [27].

**Coupling metrics:**

Coupling relations increase complexity and reduce encapsulation.

**Coupling Between Objects (CBO)**

CBO for a class is the count of the number of other classes to which it is coupled. Two classes are coupled when methods declared in one class use methods or instance variables defined by the other class [35].

**Data Abstraction Coupling (DAC)**

Data Abstraction is a technique of creating new data types suited for an application to be programmed. Data Abstraction Coupling (DAC) is defined as the number of ADTs defined in a class [63].

**Message passing Coupling (MPC)**

Message Passing Coupling (MPC) [63] is defined as the number of send statements defined in a class. If two different methods in a class C access the same method in class D, then MPC = 2.

**Response for a Class (RFC)**

The response for a class (RFC) is defined as the set of methods that can be executed in response to a message received by an object of that class [35].
<table>
<thead>
<tr>
<th>S.No</th>
<th>Metric</th>
<th>Object Oriented Attribute</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Line of code per class (LOC)</td>
<td>Class</td>
<td>[63]</td>
</tr>
<tr>
<td>2</td>
<td>Number of Attributes per Class (NOA)</td>
<td>Class</td>
<td>[63]</td>
</tr>
<tr>
<td>3</td>
<td>Number of Methods per Class (NOM)</td>
<td>Class</td>
<td>[63]</td>
</tr>
<tr>
<td>4</td>
<td>Weighted Methods per Class (WMC)</td>
<td>Class</td>
<td>[35]</td>
</tr>
<tr>
<td>5</td>
<td>Response for Class (RFC)</td>
<td>Coupling</td>
<td>[35]</td>
</tr>
<tr>
<td>6</td>
<td>Coupling between Objects (CBO)</td>
<td>Coupling</td>
<td>[35]</td>
</tr>
<tr>
<td>7</td>
<td>Data Abstraction Coupling (DAC)</td>
<td>Coupling</td>
<td>[63]</td>
</tr>
<tr>
<td>8</td>
<td>Message Passing Coupling (MPC)</td>
<td>Coupling</td>
<td>[63]</td>
</tr>
<tr>
<td>9</td>
<td>Tight Class Cohesion (TCC)</td>
<td>Cohesion</td>
<td>[27]</td>
</tr>
<tr>
<td>10</td>
<td>Information Flow based Cohesion (ICH)</td>
<td>Cohesion</td>
<td>[93]</td>
</tr>
<tr>
<td>11</td>
<td>Lack of Cohesion (LCOM)</td>
<td>Cohesion</td>
<td>[35]</td>
</tr>
<tr>
<td>12</td>
<td>Depth of Inheritance (DIT)</td>
<td>Inheritance</td>
<td>[35]</td>
</tr>
<tr>
<td>13</td>
<td>Number of Children (NOC)</td>
<td>Inheritance</td>
<td>[35]</td>
</tr>
<tr>
<td>14</td>
<td>Number of Methods Overridden by a subclass (NMO)</td>
<td>Polymorphism</td>
<td>[63]</td>
</tr>
</tbody>
</table>

*Table 6.1: Object Oriented Metrics*
Inheritance Metrics

This section discusses two different inheritance metrics, which are considered for this study.

Depth of Inheritance Tree (DIT)

The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes [35].

Number of Children (NOC)

The NOC is the number of immediate children of a class in the class hierarchy [35].

Polymorphism Metrics

Polymorphism is the characteristic of object oriented software through which the implementation of a given operation depends on the object that contains the operation.

Number of Methods Overridden by a subclass (NMO)

When a method in a child class has the same name and signature as in its parent class, then the method in the parent class is said to be overridden by the method in the child class [63].

6.4.2 Test metrics

The three test metrics used for this study are: 1) lines of code for test class (TLOC); 2) number of test methods (TM); and 3) number of test cases or asserts (TC). These metrics are calculated from the JUnit test class. Out of the three metrics mentioned previously, two (TLOC and TC) were proposed by Bruntink and Deursen [32]. The authors [124] proposed the third test metric (TM). The test metrics are defined as:
• **TLOC metric**: TLOC is defined as the number of lines of code (non comment and nonblank) in a JUnit test class. In Figure 6.2, calculatorTest class has a TLOC = 12.

• **TM metric**: TM is defined as the number of test methods in the JUnit test class. In Figure 6.2, there are two test methods in calculatorTest class: testAdd and testSubtract. Hence, for calculatorTest class the TM = 2.

• **TC metric**: TC is defined as the number of asserts in the test class. In Figure 6.2, there are two asserts in the calculatorTest class. Hence, for this class the value of TC = 2.

### 6.4.3 Goal and Hypotheses

This study evaluates the object oriented metrics using the framework proposed by Basili [28]. The goal, perspective, and environment for this study are:

• **Goal**: To judge the capability of the object oriented metrics to assess the class testability.

• **Perspective**: The testing level considered is unit level, that is, the class level. Thus, the authors judge if the object oriented metrics can assess the class testability at the unit testing level.

• **Environment**: This study is based on Java systems, which are tested at the unit testing level using the JUnit testing framework. These systems are executed using the Eclipse IDE. Clover [73] is the Eclipse plugin, which is used to calculate the code coverage of the Java systems under study. The authors formulated the following hypotheses that are tested by this study:

• **H0(c,t)**: There is no correlation between object oriented design metric c and test metric t.
• **H1(c,t):** There is a correlation between object oriented design metric c and test metric t,

where c belongs to the set of object oriented design metrics and t belongs to the set of test metrics.

### 6.4.4 Statistical Analysis

To evaluate the above hypotheses, the authors calculated the Spearman’s rank-order correlation coefficient (rs) between object oriented metrics of the java classes and the test metrics of the corresponding test classes [120]. Spearman’s rank-order correlation between object oriented metric c and test metric t is denoted by rs(c,t). The statistical significance of the correlation is also calculated. Spearman’s rank-order correlation coefficient measures the association between two variables, which are measured on the ordinal scale [114]. This kind of correlation is used because it does not depend upon the underlying data distribution. The value of rs ranges from -1 to 1. A value of 1 indicates a perfect positive correlation and -1 indicates a perfect negative correlation. A value of 0

<table>
<thead>
<tr>
<th>Java System</th>
<th>Java classes</th>
<th>JUnit classes</th>
<th>Used Classes</th>
<th>Code coverage</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lucene</td>
<td>234</td>
<td>143</td>
<td>62</td>
<td>80%</td>
<td>[70]</td>
</tr>
<tr>
<td>DarkStar</td>
<td>321</td>
<td>47</td>
<td>24</td>
<td>85%</td>
<td>[71]</td>
</tr>
<tr>
<td>HttpUnit</td>
<td>195</td>
<td>74</td>
<td>20</td>
<td>87%</td>
<td>[75]</td>
</tr>
<tr>
<td>JfreeChart</td>
<td>544</td>
<td>367</td>
<td>311</td>
<td>77%</td>
<td>[69]</td>
</tr>
</tbody>
</table>

**Table 6.2: Measurements of the systems under study**
indicates that there is no correlation. If the value of the significance for the correlation coefficient is <0.05, then the correlation coefficient is said to be significant. Based on the correlation value and the statistical significance of the correlation, the authors reject the hypothesis H0(c,t) and accept H1(c,t) with a level of confidence equal to 1-sig. To calculate rs, there should be a corresponding test class for every java class. An example of a JUnit test class (calculatorTest.java) is shown in Figure 6.2. In the four java systems, the authors used only those java classes that have their corresponding test classes. The values of object oriented metrics for java classes are paired with the values of test metrics of their corresponding test classes and then the Spearman’s rank-order correlation coefficient is calculated for the resulting pairs.

6.5 Empirical Data Collection

This study used four large open source java systems: Lucene, DarkStar, HttpUnit, and JFreeChart, of which the first is an Apache project. All the systems are unit tested at the class level by means of JUnit. The size measurements, code coverage, and the source of all the systems are given in Table 6.2. Following is a brief description of these systems:

- **Lucene**: Lucene is an open-source project written in Java, which is a text search engine library.

- **DarkStar**: Darkstar is a project which is used by developers of online games and networking applications.

- **HttpUnit**: HttpUnit is an open-source testing framework written in Java. It is used to perform the testing of the web sites without a web browser. HttpUnit can also be used with JUnit to write tests.
• **JfreeChart**: Jfreechart is an open-source Java chart library that helps developers to show quality charts in their applications.

To collect the data from the aforementioned systems, we have used the Eclipse platform and an Eclipse plugin is developed to calculate the set of metrics used.

### 6.6 Analysis Results

The results of this study are given in the Tables 6.3, 6.4, 6.5, and 6.6. The Tables contain the values of Spearman’s rank-order correlation coefficient between the object oriented design metrics and the test metrics. These Tables also contain the statistical significance value of the correlation. The number of data points is equal to the number of used classes as shown in Table 6.2. Lucene has 62 data points, DarkStar has 24 data points, HttpUnit has 20 data points, and JfreeChart has 311 data points. The design metrics that are not significantly correlated to the test metrics at the 95 percent confidence level are set in boldface and underlined in the Tables 6.3, 6.4, 6.5, and 6.6. Based on these values the authors reject $H_0(c,t)$ and accept $H_1(c,t)$. First, it is observed that some of the design metrics are highly correlated among themselves, as shown in Tables 6.7, 6.8, 6.9, and 6.10. Second, the test metrics are also correlated among themselves, as shown in the Tables 6.11, 6.12, 6.13, and 6.14. Third, some design metrics are better predictors of TLOC than TC, as shown in the Tables 6.3, 6.4, 6.5, and 6.6 which, means that more effort is spent in constructing the test cases. The design metrics that predict TLOC better than TC are test case construction factors and the others are test case generation factors [32].
<table>
<thead>
<tr>
<th></th>
<th>TLOC</th>
<th>TM</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>Corr.</td>
<td>.444</td>
<td>.377</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.000</td>
<td>.003</td>
</tr>
<tr>
<td>NOA</td>
<td>Corr.</td>
<td>.424</td>
<td>.440</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.002</td>
<td>.000</td>
</tr>
<tr>
<td>NOM</td>
<td>Corr.</td>
<td>.414</td>
<td>.450</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.004</td>
<td>.000</td>
</tr>
<tr>
<td>WMC</td>
<td>Corr.</td>
<td>.425</td>
<td>.350</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.001</td>
<td>.005</td>
</tr>
<tr>
<td>RFC</td>
<td>Corr.</td>
<td>.469</td>
<td>.379</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.000</td>
<td>.003</td>
</tr>
<tr>
<td>CBO</td>
<td>Corr.</td>
<td>.393</td>
<td>.23</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.002</td>
<td>.0121</td>
</tr>
<tr>
<td>DAC</td>
<td>Corr.</td>
<td>.431</td>
<td>.390</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.001</td>
<td>.002</td>
</tr>
<tr>
<td>MPC</td>
<td>Corr.</td>
<td>.439</td>
<td>.234</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.001</td>
<td>.0103</td>
</tr>
<tr>
<td>TCC</td>
<td>Corr.</td>
<td>-0.111</td>
<td>-0.033</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.496</td>
<td>0.912</td>
</tr>
<tr>
<td>ICH</td>
<td>Corr.</td>
<td>.329</td>
<td>.313</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.012</td>
<td>.017</td>
</tr>
<tr>
<td>LCOM</td>
<td>Corr.</td>
<td>.295</td>
<td>.332</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.026</td>
<td>.008</td>
</tr>
<tr>
<td>DIT</td>
<td>Corr.</td>
<td>-0.237</td>
<td>-.359</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.082</td>
<td>0.013</td>
</tr>
<tr>
<td>NOC</td>
<td>Corr.</td>
<td>.261</td>
<td>.179</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>.045</td>
<td>.158</td>
</tr>
<tr>
<td>NMO</td>
<td>Corr.</td>
<td>-0.177</td>
<td>-0.147</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.072</td>
<td>0.012</td>
</tr>
</tbody>
</table>

Table 6.3: Spearman’s rank order correlation coefficient and significance for Lucene

115
The following subsections provide a detailed discussion about the correlation between the design metrics and the test metrics. Through these values the impact of the design metrics on the testability of object oriented systems can be explained. The discussion of these results is provided by using the two types of factors: test case generation factors and test case construction factors.

6.6.1 Size-Related Metrics

There are four size-related metrics: lines of code, number of attributes, number of methods per class, and weighted methods per class. It is expected that a class with more lines of code will require a large test class (that is, a high TLOC). The values in the Tables 6.3, 6.4, 6.5, and 6.6 show that the four metrics are correlated with all the test metrics in all the systems.

- **Lines of code (LOC) per class**: The size of a class is a test case generation factor because a large class will require a large number of test cases (that is, a high TC). Hence, LOC is a test case generation factor. A large class can be hard to test, because its complexity increases due to the interdependencies inside the class (that is, a high TLOC), which makes it a test case construction factor. LOC is significantly correlated to all the test metrics in all the systems. Hence, high LOC increases the testing effort and decreases the testability.

- **Number of attributes (NOA)**: The attributes of a class should be initialized during testing. This means NOA should be better correlated with TLOC than with TC. Thus, NOA should be a test construction factor because more the number of attributes, the more initialization would be required, and TLOC would be high. The results show that there is a significant correlation between NOA and all the test metrics for the Lucene system. In Darkstar and HttpUnit, also there is a
correlation between NOA and the test metrics, but the correlation with TC is not
significant at the 0.05 level. In JfreeChart, the correlation between NOA and TM
is not significant at the 0.05 level. Since, there is no consistent correlation between
NOA and the test metrics in all the systems, no conclusion can be drawn regarding
the relationship of testability with NOA. Further research is required to explore
the relationship between NOA and the testability.

- **Number of methods (NOM):** NOM is a test case generation factor. It is not the
test case construction factor, as the number of methods only affects the number of
test cases to be written but not the complexity of writing them. This means that
NOM predicts TLOC and TC equally well. As expected, all the four systems
(Lucene, HttPUnit, Darkstar, and JfreeChart) predict TLOC and TC equally.
Hence, as NOM increases, the testing effort increases and the testability decreases.

- **Weighted methods per class (WMC):** WMC is a count of the sum of the
complexities of all the methods in a class. The WMC metric is strongly correlated
to NOM in all the four systems. Also, the correlation values of other metrics are
similar for both the NOM and WMC. Hence, the explanation for the correlation of
WMC with the test metrics is similar to the one given for NOM. WMC is a test
case generation factor and not a test case construction factor.

### 6.6.2 Coupling Metrics

There are four coupling metrics analyzed in this work and these are discussed
below.

- **Response for class (RFC):** RFC is the set of methods that can be executed in
response to a message received by an object of that class. The NOM metric is an
<table>
<thead>
<tr>
<th>DarkStar</th>
<th>TLOC</th>
<th>TM</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>Corr.</td>
<td>.521</td>
<td>.490</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.009</td>
<td>0.015</td>
</tr>
<tr>
<td>NOA</td>
<td>Corr.</td>
<td>.501</td>
<td>.460</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.012</td>
<td>0.023</td>
</tr>
<tr>
<td>NOM</td>
<td>Corr.</td>
<td>.627</td>
<td>.584</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>WMC</td>
<td>Corr.</td>
<td>.519</td>
<td>.509</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.009</td>
<td>0.010</td>
</tr>
<tr>
<td>RFC</td>
<td>Corr.</td>
<td>.609</td>
<td>.585</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>CBO</td>
<td>Corr.</td>
<td>.667</td>
<td>.605</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.0003</td>
<td>0.0017</td>
</tr>
<tr>
<td>DAC</td>
<td>Corr.</td>
<td>.621</td>
<td>.546</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.0012</td>
<td>0.005</td>
</tr>
<tr>
<td>MPC</td>
<td>Corr.</td>
<td>.578</td>
<td>.572</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.003</td>
<td>0.003</td>
</tr>
<tr>
<td>TCC</td>
<td>Corr.</td>
<td><strong>-0.122</strong></td>
<td><strong>-0.146</strong></td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td><strong>0.57</strong></td>
<td><strong>0.49</strong></td>
</tr>
<tr>
<td>ICH</td>
<td>Corr.</td>
<td>.434</td>
<td>.448</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.033</td>
<td>0.028</td>
</tr>
<tr>
<td>LCOM</td>
<td>Corr.</td>
<td>.522</td>
<td>.466</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.008</td>
<td>0.021</td>
</tr>
<tr>
<td>DIT</td>
<td>Corr.</td>
<td>.632</td>
<td>.505</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.000</td>
<td>0.011</td>
</tr>
<tr>
<td>NOC</td>
<td>Corr.</td>
<td><strong>0.099</strong></td>
<td><strong>0.071</strong></td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td><strong>0.64</strong></td>
<td><strong>0.74</strong></td>
</tr>
<tr>
<td>NMO</td>
<td>Corr.</td>
<td>.631</td>
<td>.650</td>
</tr>
<tr>
<td></td>
<td>Sig.</td>
<td>0.0009</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 6.4: Spearman’s rank order correlation coefficient and significance for DarkStar
<table>
<thead>
<tr>
<th>HttpUnit</th>
<th>TLOC</th>
<th>TM</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>0.62</td>
<td>0.596</td>
<td>0.595</td>
</tr>
<tr>
<td></td>
<td>0.009</td>
<td>0.013</td>
<td>0.023</td>
</tr>
<tr>
<td>NOA</td>
<td>0.579</td>
<td>0.513</td>
<td>0.465</td>
</tr>
<tr>
<td></td>
<td>0.010</td>
<td>0.035</td>
<td><strong>0.082</strong></td>
</tr>
<tr>
<td>NOM</td>
<td>0.608</td>
<td>0.651</td>
<td>0.647</td>
</tr>
<tr>
<td></td>
<td>0.009</td>
<td>0.006</td>
<td>0.014</td>
</tr>
<tr>
<td>WMC</td>
<td>0.585</td>
<td>0.572</td>
<td>0.572</td>
</tr>
<tr>
<td></td>
<td>0.017</td>
<td>0.021</td>
<td>0.038</td>
</tr>
<tr>
<td>RFC</td>
<td>0.622</td>
<td>0.627</td>
<td>0.618</td>
</tr>
<tr>
<td></td>
<td>0.009</td>
<td>0.007</td>
<td>0.017</td>
</tr>
<tr>
<td>CBO</td>
<td>0.637</td>
<td>0.664</td>
<td>0.594</td>
</tr>
<tr>
<td></td>
<td>0.002</td>
<td>0.000</td>
<td>0.001</td>
</tr>
<tr>
<td>DAC</td>
<td>0.405</td>
<td>0.662</td>
<td>0.745</td>
</tr>
<tr>
<td></td>
<td><strong>0.105</strong></td>
<td>0.007</td>
<td>0.002</td>
</tr>
<tr>
<td>MPC</td>
<td>0.587</td>
<td>0.62</td>
<td>0.543</td>
</tr>
<tr>
<td></td>
<td>0.005</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>TCC</td>
<td><strong>-0.46</strong></td>
<td>-0.56</td>
<td>-0.54</td>
</tr>
<tr>
<td></td>
<td><strong>0.056</strong></td>
<td>0.026</td>
<td>0.036</td>
</tr>
<tr>
<td>ICH</td>
<td>0.511</td>
<td>0.432</td>
<td>0.481</td>
</tr>
<tr>
<td></td>
<td>0.040</td>
<td>0.0100</td>
<td>0.046</td>
</tr>
<tr>
<td>LCOM</td>
<td>0.382</td>
<td>0.475</td>
<td>0.428</td>
</tr>
<tr>
<td></td>
<td>0.0151</td>
<td>0.049</td>
<td>0.0192</td>
</tr>
<tr>
<td>DIT</td>
<td>0.54</td>
<td>0.553</td>
<td>0.616</td>
</tr>
<tr>
<td></td>
<td>0.027</td>
<td>0.023</td>
<td>0.008</td>
</tr>
<tr>
<td>NOC</td>
<td>0.234</td>
<td>0.3</td>
<td>0.37</td>
</tr>
<tr>
<td></td>
<td><strong>0.380</strong></td>
<td><strong>0.395</strong></td>
<td><strong>0.315</strong></td>
</tr>
<tr>
<td>NMO</td>
<td>0.36</td>
<td>0.4</td>
<td>0.379</td>
</tr>
<tr>
<td></td>
<td>0.0150</td>
<td>0.042</td>
<td>0.0143</td>
</tr>
</tbody>
</table>

Table 6.5: Spearman’s rank order correlation coefficient and significance for HttpUnit
<table>
<thead>
<tr>
<th>JfreeChart</th>
<th>TLOC</th>
<th>TM</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>0.557</td>
<td>0.388</td>
<td>0.485</td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>NOA</td>
<td>0.306</td>
<td><strong>0.057</strong></td>
<td>0.342</td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td><strong>0.316</strong></td>
<td>0.000</td>
</tr>
<tr>
<td>NOM</td>
<td>0.584</td>
<td>0.372</td>
<td>0.587</td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>WMC</td>
<td>0.574</td>
<td>0.387</td>
<td>0.522</td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>RFC</td>
<td>0.599</td>
<td>0.455</td>
<td>0.528</td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>CBO</td>
<td>0.386</td>
<td>0.365</td>
<td>0.262</td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>DAC</td>
<td><strong>0.12</strong></td>
<td>-0.04</td>
<td><strong>0.162</strong></td>
</tr>
<tr>
<td></td>
<td>0.034</td>
<td><strong>0.455</strong></td>
<td>0.004</td>
</tr>
<tr>
<td>MPC</td>
<td>0.447</td>
<td>0.389</td>
<td>0.334</td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>TCC</td>
<td>-0.08</td>
<td>-0.03</td>
<td>-0.11</td>
</tr>
<tr>
<td></td>
<td>0.171</td>
<td><strong>0.572</strong></td>
<td>0.047</td>
</tr>
<tr>
<td>lCH</td>
<td>0.468</td>
<td>0.342</td>
<td>0.402</td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>LCOM</td>
<td>0.383</td>
<td>0.222</td>
<td>0.401</td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>DIT</td>
<td>0.301</td>
<td>0.406</td>
<td><strong>0.192</strong></td>
</tr>
<tr>
<td></td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
</tr>
<tr>
<td>NOC</td>
<td><strong>0.186</strong></td>
<td><strong>0.143</strong></td>
<td><strong>0.192</strong></td>
</tr>
<tr>
<td></td>
<td>0.001</td>
<td>0.012</td>
<td>0.001</td>
</tr>
<tr>
<td>NMO</td>
<td><strong>0.109</strong></td>
<td><strong>0.148</strong></td>
<td><strong>0.003</strong></td>
</tr>
<tr>
<td></td>
<td>0.055</td>
<td>0.009</td>
<td><strong>0.956</strong></td>
</tr>
</tbody>
</table>

Table 6.6: Spearman’s rank order correlation coefficient and significance for JfreeChart
important component of the RFC metric. This fact also is reflected by the strong
correlation between RFC and NOM metrics (see Tables 6.7-6.10). The correlation
between RFC and all the test metrics is high. RFC is a test case construction factor
because it is better correlated to TLOC than TC in all the systems. This metric is
significantly correlated to all the test metrics in all the systems, which means that
RFC can predict the testing effort. Thus, as RFC increases, the testing effort
increases and the testability decreases.

- **Coupling between objects (CBO):** CBO for a class is the count of the number
  of other classes to which it is coupled. Theoretically, an increase in CBO should
  result in an increase in the testing effort. The same is shown by the values in the
  Tables 6.3, 6.4, 6.5, and 6.6. There is a significant correlation between CBO and
  all the test metrics for all the systems. Hence, an increase in CBO increases the
testing effort and lowers the testability.

- **Data abstraction coupling (DAC):** DAC measures the number of abstract data
  types defined in a class. Hence, in the actual program the objects of these abstract
  data types have to be initialized before they are used. The initialization of objects
  leads to the initialization of the attributes of the class. This means that the
correlation of DAC with TLOC should be higher than TC. The results show that in
Lucene, the DAC metric is significantly correlated to TLOC and TM and
insignificantly correlated to TC. In Darkstar, the correlation is significant with all
the test metrics. In HttpUnit, this metric is significantly correlated to TM and TC
but insignificantly correlated to TLOC. In JFreeChart, there is no significant
correlation of DAC with any of the test metrics. This can be explained from the
correlation of DAC with NOA. There is a strong correlation between DAC and
NOA in Darkstar and it explains the correlation of DAC with the test metrics in
Darkstar. In Lucene, the correlation between DAC and NOA is moderate; hence, there is a correlation between DAC with TLOC and TM. In HtpUnit, the correlation between DAC and NOA is low. Hence, the correlation of DAC is insignificant with TLOC and significant with the other two test metrics. However, the correlation between DAC and NOA is low in JfreeChart. Hence, the authors conclude that due to the inconsistent correlation between DAC and the test metrics of different systems, no general conclusions can be drawn regarding the relationship of DAC with the testing effort and with testability.

• **Message passing coupling (MPC):** MPC measures the number of method calls defined in the methods of a class to the methods in other classes. It is significantly correlated to all the test metrics in all the four systems. From the definition of MPC, it is clear that the number of methods is a major component of the MPC metric. The same is shown by the correlation between MPC and NOM (Tables 6.7-6.10). Also, MPC is a test case construction factor because it is better correlated to TLOC than TC in all the systems. The authors conclude that as MPC increases, the testing effort increases and the testability decreases.

From the previous discussion, the authors conclude that the testing effort is related to the three coupling metrics CBO, RFC, and MPC. As the value of these metrics increases, the testing effort increases, and thus, it lowers the testability of the system. No conclusions, however, can be drawn regarding the relationship of the DAC metric and the testability.

### 6.6.3 Cohesion Metrics

There are three cohesion metrics analyzed in this work and these are discussed below.
### Table 6.7: Correlation of design metrics of Lucene

<table>
<thead>
<tr>
<th>Lucene</th>
<th>LOC</th>
<th>NOA</th>
<th>NOM</th>
<th>WMC</th>
<th>RFC</th>
<th>CBO</th>
<th>DAC</th>
<th>MPC</th>
<th>TCC</th>
<th>ICH</th>
<th>LCOM</th>
<th>DIT</th>
<th>NOC</th>
<th>NMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOA</td>
<td>.728</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOM</td>
<td>.774</td>
<td>.721</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WMC</td>
<td>.962</td>
<td>.721</td>
<td>.797</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFC</td>
<td>.814</td>
<td>.622</td>
<td>.794</td>
<td>.830</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBO</td>
<td>.497</td>
<td>.421</td>
<td>.296</td>
<td>.467</td>
<td>.629</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAC</td>
<td>.420</td>
<td>.581</td>
<td>.488</td>
<td>.383</td>
<td>.493</td>
<td>.477</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPC</td>
<td>.586</td>
<td>.490</td>
<td>.416</td>
<td>.573</td>
<td>.753</td>
<td>.889</td>
<td>.514</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCC</td>
<td>0.035</td>
<td>0.17</td>
<td>0.043</td>
<td>0.033</td>
<td>0.029</td>
<td>0</td>
<td>0.16</td>
<td>0.052</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH</td>
<td>.712</td>
<td>.421</td>
<td>.720</td>
<td>.729</td>
<td>.688</td>
<td>.226</td>
<td>.268</td>
<td>.284</td>
<td>-.216</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCOM</td>
<td>.680</td>
<td>.845</td>
<td>.736</td>
<td>.673</td>
<td>.548</td>
<td>.26</td>
<td>.434</td>
<td>.345</td>
<td>.033</td>
<td>.443</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIT</td>
<td>-.300</td>
<td>-.313</td>
<td>-.380</td>
<td>-.244</td>
<td>-.225</td>
<td>-.029</td>
<td>-.23</td>
<td>.042</td>
<td>.181</td>
<td>-.392</td>
<td>-.345</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOC</td>
<td>-.095</td>
<td>-.04</td>
<td>.04</td>
<td>-.09</td>
<td>-.025</td>
<td>-.155</td>
<td>.006</td>
<td>-.119</td>
<td>.087</td>
<td>.124</td>
<td>.02</td>
<td>-.167</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>NMO</td>
<td>0.077</td>
<td>0.096</td>
<td>0.15</td>
<td>0.167</td>
<td>0.147</td>
<td>0.12</td>
<td>-.092</td>
<td>0.117</td>
<td>.323</td>
<td>-.064</td>
<td>.126</td>
<td>.396</td>
<td>-.219</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 6.8: Correlation of design metrics of DarkStar

<table>
<thead>
<tr>
<th>DarkStar</th>
<th>LOC</th>
<th>NOA</th>
<th>NOM</th>
<th>WMC</th>
<th>RFC</th>
<th>CBO</th>
<th>DAC</th>
<th>MPC</th>
<th>TCC</th>
<th>ICH</th>
<th>LCOM</th>
<th>DIT</th>
<th>NOC</th>
<th>NMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOA</td>
<td>.704</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOM</td>
<td>.905</td>
<td>0.699</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WMC</td>
<td>.982</td>
<td>0.654</td>
<td>0.921</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFC</td>
<td>.897</td>
<td>0.74</td>
<td>0.938</td>
<td>0.901</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBO</td>
<td>.639</td>
<td>0.698</td>
<td>0.618</td>
<td>0.602</td>
<td>0.762</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAC</td>
<td>.73</td>
<td>0.875</td>
<td>0.699</td>
<td>0.673</td>
<td>0.788</td>
<td>0.866</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPC</td>
<td>.813</td>
<td>0.779</td>
<td>0.773</td>
<td>0.803</td>
<td>0.869</td>
<td>0.86</td>
<td>0.87</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCC</td>
<td>-.198</td>
<td>-.165</td>
<td>-.145</td>
<td>-.236</td>
<td>-.158</td>
<td>-.175</td>
<td>-.199</td>
<td>-.359</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH</td>
<td>.829</td>
<td>0.404</td>
<td>0.762</td>
<td>0.865</td>
<td>0.734</td>
<td>0.479</td>
<td>0.428</td>
<td>0.595</td>
<td>-.083</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCOM</td>
<td>.602</td>
<td>0.681</td>
<td>0.639</td>
<td>0.565</td>
<td>0.663</td>
<td>0.542</td>
<td>0.672</td>
<td>0.57</td>
<td>-.191</td>
<td>0.379</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIT</td>
<td>.623</td>
<td>0.822</td>
<td>0.753</td>
<td>0.593</td>
<td>0.771</td>
<td>0.761</td>
<td>0.851</td>
<td>0.752</td>
<td>-.126</td>
<td>0.282</td>
<td>0.602</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOC</td>
<td>.35</td>
<td>0.317</td>
<td>0.44</td>
<td>0.386</td>
<td>0.461</td>
<td>0.145</td>
<td>0.254</td>
<td>0.374</td>
<td>-.188</td>
<td>0.174</td>
<td>0.285</td>
<td>0.355</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>NMO</td>
<td>0.526</td>
<td>0.547</td>
<td>0.687</td>
<td>0.522</td>
<td>0.631</td>
<td>0.403</td>
<td>0.472</td>
<td>0.572</td>
<td>0.005</td>
<td>0.357</td>
<td>0.589</td>
<td>0.587</td>
<td>0.247</td>
<td>1</td>
</tr>
<tr>
<td>Unit</td>
<td>LOC</td>
<td>NOA</td>
<td>NOM</td>
<td>WMC</td>
<td>RFC</td>
<td>CBO</td>
<td>DAC</td>
<td>MPC</td>
<td>TCC</td>
<td>ICH</td>
<td>LCOM</td>
<td>DIT</td>
<td>NOC</td>
<td>NMO</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>LOC</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOA</td>
<td>0.842</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOM</td>
<td>0.962</td>
<td>0.755</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WMC</td>
<td>0.987</td>
<td>0.842</td>
<td>0.975</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFC</td>
<td>0.975</td>
<td>0.786</td>
<td>0.987</td>
<td>0.977</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBO</td>
<td>0.574</td>
<td>0.474</td>
<td>0.572</td>
<td>0.538</td>
<td>0.628</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAC</td>
<td>0.438</td>
<td>0.323</td>
<td>0.556</td>
<td>0.43</td>
<td>0.504</td>
<td>0.504</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPC</td>
<td>0.601</td>
<td>0.47</td>
<td>0.608</td>
<td>0.57</td>
<td>0.665</td>
<td>0.985</td>
<td>0.495</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCC</td>
<td>-0.23</td>
<td>-0.18</td>
<td>-0.25</td>
<td>-0.169</td>
<td>-0.203</td>
<td>-0.168</td>
<td>-0.556</td>
<td>-0.107</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH</td>
<td>0.867</td>
<td>0.742</td>
<td>0.831</td>
<td>0.88</td>
<td>0.832</td>
<td>0.317</td>
<td>0.27</td>
<td>0.313</td>
<td>-0.203</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCOM</td>
<td>0.779</td>
<td>0.68</td>
<td>0.809</td>
<td>0.838</td>
<td>0.778</td>
<td>0.434</td>
<td>0.393</td>
<td>0.458</td>
<td>-0.071</td>
<td>0.727</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIT</td>
<td>0.606</td>
<td>0.347</td>
<td>0.615</td>
<td>0.544</td>
<td>0.619</td>
<td>0.481</td>
<td>0.529</td>
<td>0.513</td>
<td>-0.382</td>
<td>0.349</td>
<td>0.164</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOC</td>
<td>0.128</td>
<td>0.034</td>
<td>0.226</td>
<td>0.153</td>
<td>0.169</td>
<td>-0.187</td>
<td>0.307</td>
<td>-0.233</td>
<td>-0.552</td>
<td>0.29</td>
<td>0.099</td>
<td>0.072</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>NMO</td>
<td>0.68</td>
<td>0.33</td>
<td>0.753</td>
<td>0.697</td>
<td>0.722</td>
<td>0.357</td>
<td>0.239</td>
<td>0.411</td>
<td>-0.151</td>
<td>0.548</td>
<td>0.571</td>
<td>0.602</td>
<td>0.074</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 6.9: Correlation of design metrics of HttpUnit**

- **Tight class cohesion (TCC):** TCC is not significantly correlated to the test metrics in the following three systems: Lucene, DarkStar, and JFreeChart. In HttpUnit, there is a negative correlation between TCC and all the test metrics. Hence, no conclusions can be drawn for TCC.

- **Lack of cohesion of methods (LCOM):** The results show that LCOM is significantly correlated to all the three test metrics in all the systems. Hence, the testing effort increases with a high LCOM value and leads to lower testability.

**Information based Cohesion (ICH):** The results show that this metric is significantly correlated to all the test metrics in all the systems. ICH is a test case construction factor (High TLOC than TC). Hence testing effort increases with increase in ICH.
Table 6.10: Correlation of design metrics of JFreeChart

<table>
<thead>
<tr>
<th></th>
<th>LOC</th>
<th>NOA</th>
<th>NOM</th>
<th>WMC</th>
<th>RFC</th>
<th>CBO</th>
<th>DAC</th>
<th>MPC</th>
<th>TCC</th>
<th>ICH</th>
<th>LCOM</th>
<th>DIT</th>
<th>NOC</th>
<th>NMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOA</td>
<td>0.678</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOM</td>
<td>0.835</td>
<td>0.689</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WMC</td>
<td>0.975</td>
<td>0.683</td>
<td>0.896</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFC</td>
<td>0.911</td>
<td>0.555</td>
<td>0.885</td>
<td>0.917</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBO</td>
<td>0.702</td>
<td>0.292</td>
<td>0.482</td>
<td>0.624</td>
<td>0.771</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAC</td>
<td>0.308</td>
<td>0.421</td>
<td>0.348</td>
<td>0.333</td>
<td>0.340</td>
<td>0.218</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPC</td>
<td>0.735</td>
<td>0.297</td>
<td>0.547</td>
<td>0.680</td>
<td>0.824</td>
<td>0.937</td>
<td>0.260</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCC</td>
<td>-0.116</td>
<td>0.031</td>
<td>-0.036</td>
<td>-0.07</td>
<td>-0.150</td>
<td>-0.235</td>
<td>0.058</td>
<td>-0.240</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICH</td>
<td>0.667</td>
<td>0.338</td>
<td>0.611</td>
<td>0.699</td>
<td>0.683</td>
<td>0.435</td>
<td>0.355</td>
<td>0.518</td>
<td>-0.171</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCOM</td>
<td>0.708</td>
<td>0.689</td>
<td>0.724</td>
<td>0.707</td>
<td>0.699</td>
<td>0.475</td>
<td>0.279</td>
<td>0.491</td>
<td>-0.374</td>
<td>0.498</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIT</td>
<td>0.346</td>
<td>0.006</td>
<td>0.290</td>
<td>0.307</td>
<td>0.494</td>
<td>0.561</td>
<td>-0.037</td>
<td>0.569</td>
<td>-0.167</td>
<td>0.249</td>
<td>0.285</td>
<td>1.000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOC</td>
<td>0.279</td>
<td>0.254</td>
<td>0.301</td>
<td>0.288</td>
<td>0.309</td>
<td>0.213</td>
<td>0.145</td>
<td>0.210</td>
<td>-0.143</td>
<td>0.244</td>
<td>0.291</td>
<td>0.047</td>
<td>1.000</td>
<td></td>
</tr>
<tr>
<td>NMO</td>
<td>0.321</td>
<td>0.291</td>
<td>0.346</td>
<td>0.355</td>
<td>0.321</td>
<td>0.169</td>
<td>0.175</td>
<td>0.141</td>
<td>0.366</td>
<td>0.269</td>
<td>0.193</td>
<td>0.181</td>
<td>0.062</td>
<td>1.000</td>
</tr>
</tbody>
</table>

6.6.4 Inheritance-Related Metrics

There are two inheritance metrics analyzed in this work and these are discussed below.

- **Depth of inheritance (DIT):** DIT measures the number of parent classes in a class. There is an inverse relationship between DIT and test metrics in Lucene, which should not be the case, and a significant correlation among DIT and test metrics in Darkstar and HttpUnit. Also, there is a significant correlation between DIT and the test metrics TLOC and TM in JFreeChart. This can be explained based on the fact that as the DIT increases, the testing effort increases because inherited methods are retested in the derived class. This fact is shown by the
correlation values among DIT and the test metrics in Darkstar, HttpUnit, and JfreeChart, but the same is not true for Lucene.

• **Number of children (NOC):** NOC measures the number of child classes in a class. In the case of NOC, there is no consistent correlation between NOC and test metrics in all the systems, so no conclusions can be drawn. In Lucene, NOC is correlated to only one test metric which is TLOC. In HttpUnit, there is a correlation between NOC and the test metrics, but the correlation is not significant. In Darkstar and JfreeChart there is no significant correlation of NOC with the test metrics. This can be explained based on the fact that in these systems the value of NOC was much less for a large number of classes. Hence, no significant correlation was found between NOC and test metrics for these systems.

Hence, no conclusions can be drawn regarding the relationship between inheritance-related metrics and the testability based on the correlation values.

### 6.6.5 Polymorphism-Related Metrics

There is only one polymorphism metric analyzed in this work and is discussed below.

• **Number of methods overridden by a sub class (NMO):** When a method in a child class has the same name and signature as in its parent class, the method in the parent class is said to be overridden by the method in the child class. Polymorphism increases the testing effort and hence decreases the testability. The correlation between NMO and test metrics is significant in only two systems, that is, DarkStar and HttpUnit. There is an insignificant correlation between NMO and test metrics in the case of Lucene and JfreeChart. Hence, we cannot make a conclusion about the impact of NMO on the testing effort or on the testability.
<table>
<thead>
<tr>
<th>Lucene</th>
<th>TLOC</th>
<th>TM</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLOC</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TM</td>
<td>0.751</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td>0.633</td>
<td>0.582</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.11: Correlation of Test metrics of Lucene

<table>
<thead>
<tr>
<th>Darkstar</th>
<th>TLOC</th>
<th>TM</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLOC</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TM</td>
<td>0.946</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td>0.928</td>
<td>0.909</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.12: Correlation of Test metrics of DarkStar

<table>
<thead>
<tr>
<th>HttpUnit</th>
<th>TLOC</th>
<th>TM</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLOC</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TM</td>
<td>0.9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td>0.84</td>
<td>0.9</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.13: Correlation of Test metrics of HttpUnit

<table>
<thead>
<tr>
<th>JfreeChart</th>
<th>TLOC</th>
<th>TM</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLOC</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TM</td>
<td>0.83</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TC</td>
<td>0.88</td>
<td>0.71</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.14: Correlation of Test metrics of JfreeChart
6.7 Discussion of Results

The results of this study are summarized below.

1) An increase in LOC, NOM, and WMC will have a strong increasing impact on the testing effort and thus will result in low testability. The relationship between number of attributes and testability, however, needs to be further explored.

2) As expected, an increase in coupling in classes is likely to increase the testing effort in the software. Coupling increases the complexity of the software and reduces encapsulation. Hence, for high testability, coupling should be low. Out of four coupling metrics, however, only three (CBO, MPC, and RFC) reflect this fact, and the relationship of DAC with testability could not be established.

3) LCOM is likely to increase the testing effort. On the other hand, cohesion in methods lowers the testing effort and increases the testability.

4) Logically, it seems that an increase in the TCC value should increase the testing effort and thus decrease the testability. The results of this study, however, do not show any significant effect of TCC on the testability.

5) An increase in RFC increases the testing effort and thus decrease its testability.

6) Logically, it seems that an increase in NOC and DIT should increase the testing effort and thus decreases the testability. The results of this study, however, do not show a significant effect of NOC or DIT on the testability.

7) There is a significant positive relationship between the test metrics and NMO (an indicator of polymorphism) in the two systems under study. This is true theoretically because an increase in polymorphism increases the testing effort and lowers the testability. Also, polymorphism increases the complexity of the software due to run-time binding. But this fact is not shown by the other two
systems of this study. Hence, no general conclusions can be drawn about the effect of polymorphism on the testability.

6.8 Conclusions

In this chapter, the relationship between the design metrics and the test metrics is determined and the effect of OO design metrics on the testing effort and testability is found. The relationship between the java classes and their JUnit test classes in four large Java systems was analyzed. A significant correlation was found between most of the OO design metrics and the test metrics. The results of correlation show that out of four size metrics, three metrics (LOC, NOM, and WMC) are correlated to the testing effort. It is shown that as the size of the software increases, the testability decreases because the testing effort increases. Out of four coupling metrics, three metrics (RFC, CBO, and MPC) are highly correlated to the test metrics. No conclusions are drawn about the relationship of the testability with NOA and DAC. This issue needs further research. Two cohesion metrics, ICH and LCOM, are found to be correlated to the test metrics. Hence, high values of ICH and LCOM increase the testing effort and decrease testability. TCC does not have any significant correlation with test metrics. Hence, no conclusion is drawn regarding the relationship between TCC and testability. The results of correlation between design metrics and test metrics show that there is no relationship of inheritance metrics (DIT and NOC) and polymorphism metrics (NMO) with the testing effort and the testability. This is because there was less inheritance and polymorphism used in the systems under investigation.

While research continues, practitioners can apply the aforementioned eight metrics to predict the testability of a class, which are found to be highly correlated
to the testing effort. One can assess the quality of the software in the earlier stages of software development by finding the values of these metrics. The classes found to have more testing effort will need extra attention during the rest of the development. Hence, software practitioners can use these metrics to assess the testing effort of a class in the earlier phases of software development. Also, the testing activities can be planned in advance, leading to a reduced cost of testing. The assessment of testability could be of great help for planning and executing testing activities by paying more attention to classes with low testability. These classes will need extra attention during the remainder of the development. Also, if required, developers can reconsider the design of these classes and take corrective actions.