EMPIRICAL VALIDATION OF CLASS COUPLING METRICS AS CHANGEABILITY INDICATORS IN SOFTWARE EVOLUTION

BY

Ali Abdallah Al-Zouri

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

Dhahran 31261, Saudi Arabia

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Ali Abdallah Al-Zouri** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Thesis Committee

Dr. Mahmoud Elish (Chairman)

Dr. Naser Darwish (Member)

Dr. Mohammad Alshayeb (Member)

Dr. Kanaan Faisal

Department Chairman

Dr. Salam A. Zummo

Dean of Graduate Studies

20/6/09

Date

Dedication

This thesis is dedicated to the people I love: my parents and wife

Acknowledgement

All thanks are due to Allah first and foremost of his countless blessings. Acknowledgement is due to King Fahd University of Petroleum & Minerals and ICS Department for supporting this research.

My sincere appreciation goes to Dr. Mahmoud Elish, for all assistance, advice, encouragement and invaluable support given as my advisor and mentor during my journey at KFUPM and throughout the period of this research in particular. I also wish to thank my thesis committee members, Dr. Mohammad Alshayeb and Dr. Nasir Darwish, for their help, support, and contributions.

Finally, I wish to express my appreciation to my parents and wife for being patient with me and offering words of encouragement at moments of depression.

Table of Contents

List of Tables

List of Figures

Thesis Abstract

NAME: ALI ABDALLAH AL-ZOURI

TITLE: EMPIRICAL VALIDATION OF CLASS COUPLING METRICS AS CHANGEABILITY INDICATORS IN SOFTWARE EVOLUTION

MAJOR FIELD: COMPUTER SCIENCE

DATE OF DEGREE: JANUARY 2009

During the development and maintenance of Object-Oriented (OO) software systems, the information on the classes which are more change-prone is very useful. Early knowledge of the change-prone classes can provide an easier way for developing stable software systems. In addition, this knowledge will assist software developers to focus only on those changed classes during software testing process. In this research, coupling metrics are investigated to explore their capability to identify change-prone classes and predict the change-density of classes in evolving OO software systems. Accuracy of coupling metrics models is compared with that of models built on different OO metrics. Results show that coupling metrics are good indicators of change-prone classes and change-density of classes in evolving OO software systems, and they provide comparable or even better results than models built on other OO metrics. Moreover, results show that some of the coupling metrics are still good indicators of change-prone classes and change-density of classes in evolving OO software systems even after controlling for class size.

خلاصة الرسالة

الأسم: **علي عبدالله الزوري**

عنوان الدراسة: **دراسة تطبيقيه عن مدى مقدرة مقاييس التماسك من التنبؤ للوحدات المتغيرة في البرامج متعددة الإصدارات**

التخصص: **علوم الحاسب الآلي**

تاريخ التخرج: **يناير 2009**

تعد المعرفة بالوحدات الأكثر عرضه للتغيير خلال صيانة وتطوير البرامج الحاسوبية مفيدة جداً. المعرفة المبكرة بالوحدات المتعرضة للتغيير تساهم في التوصل إلي طريق سهل لتطوير برامج حاسوب مستقرة. بالإضافة إلي أن هذه المعرفة تساعد مطوري البرامج الحاسوبية في التركيز فقط على الوحدات المتغيرة خلال إجراء اختبار البرامج الحاسوبية. تم في هذا البحث تحري مقدرة مقاييس الاقتران لاكتشاف قدرتها في التعرف على الوحدات القابلة للتغيير، وتوقع درجة التغيير في تطوير البرامج الحاسوبية. تم مقارنة دقة نماذج مقاييس الاقتران بنماذج أخرى التي تم بناؤها على مقاييس أخرى مختلفة. وقد أظهرت النتائج أن مقاييس الاقتران تعد مؤشراً جيداً للوحدات المتعرضة للتغيير، و لمعرفة درجة التغيير في الوحدات في تطور البرامج الحاسوبية، كما أن مقاييس الاقتران أعطت نتائج مشابهة أو أفضل مقارنة بالنماذج الأخرى المبنية على مقاييس البرامج الحاسوبية. علاوة على ذلك ، فإن النتائج أظهرت أن بعض مقاييس الاقتران تعتبر كمؤشر جيد للوحدات القابلة للتغيير ودرجة التغيير فيها في تطور البرامج الحاسوبية حتى بعد التحكم في حجم الوحدات.

Chapter 1

Introduction

Many metrics have been proposed in the literature to capture the quality of object-oriented (OO) software systems. Such metrics are aimed to assess the quality of software systems. One aspect of a software system which degrades its quality is the strength of association between different modules in the system. This aspect is called "coupling". Strong coupling between modules makes them highly inter-related and as a consequence they become difficult to understand, change and correct [20].

In software engineering research, increased importance is being placed on software metrics, especially on those that predict the quality of a software system, and this has led to an increased amount of research in this area. Coupling metrics are one type of software metrics that is used to evaluate and predict the quality of a software system. Stevens et al. [35], who first introduced coupling, define coupling as "the measure of the strength of association established by a connection from module to another". [20]. Low coupling has been considered as an important characteristic of good software systems. Low coupling should allow individual modules in a system to be easily modified with relatively little worry about affecting other modules in a system [20].

Validation of coupling metrics is very important in order to determine their effectiveness in practice. There are two types of validations: internal and external [16]. Internal validation is a theoretical exercise that ensures that a metric is a true numerical characterization of the property it claims to measure. External validation involves empirically demonstrating that a metric is associated with some important external metrics (such as measures of changeability or testability) [16]. Whereas a coupling metric may be correct from a theoretical perspective, it may not be of practical use in industrial settings. Metrics may be difficult to collect or may not really measure the intended quality attribute. Empirical validation is necessary to demonstrate the usefulness of a metric in practical applications. Large numbers of empirical studies correlate coupling metrics with software quality attributes such as fault-proneness [15][17][21][33], change-proneness [1][4], reusability [7], testability [23] and maintainability [34].

Few of the published studies have emphasized the validation of a comprehensive set of coupling metrics in which different mechanisms which constitute coupling are considered. Briand et al. [21] published a comprehensive study with respect to fault-proneness. Also, previous empirical studies focused on the analysis of a single release of a software system. However, most of the current software systems consist of several releases. Olague et al. [10] conducted an empirical study with some coupling metrics considering multiple releases of a system developed in iterative fashion. They conducted the study with respect to fault-proneness. Thus, there is a need for a comprehensive empirical study that considers coupling metrics according to a unified

framework and correlates them with software quality attributes throughout software evolution.

One of the important software attributes is the changeability of a system. It is the nature of a software system to be changed. Reasons for changing software systems fall into the following categories:

- Perfective: Changes are made to add new requirements to the software system
- Corrective: Changes are made to repair software defects
- Adaptive: Changes are made to keep pace with changing environment
- Preventive: Changes are made to improve future maintainability and reliability of a software system.[27]

In the ISO 9126 software quality factors, changeability is considered as a sub-attribute of maintainability. It characterizes the amount of effort required to change a system. Changeability of the software system is very important to be studied, especially in an evolving software system, in order to identity those modules which are likely to be modified in the subsequent releases of a system. In this study, coupling metrics will be investigated in two ways to determine their capability to capture the changeability of evolving software system. The first way is by identifying the change-prone classes from one release to the next release. The second way is by predicting the density of changes occurring in the classes from one release to the next.

1.1 Problem Statement

The relationship between coupling metrics and change-proneness of classes in evolving object-oriented software is unknown due to lack of empirical studies. This implies that the practical usefulness of these metrics, in this context, is unclear.

Empirical study will give evidence about the applicability of using coupling metrics as indicators of class change-proneness and change-density. Thus, there is a need to empirically investigate the capability of coupling metrics to identify the change-prone classes and predict the density of changes from one release to the next release.

1.2 Rationale: Problem Importance

Being able to identify the change-prone classes in a software system is very important. Change-prone classes increase project cost by requiring developers to spend more effort and time. Moreover, in most software systems, a great majority of changes is rooted in a small proportion of classes [1]. In other words, around 80% of the changes are actually rooted in around 20% of the classes. This phenomenon has been commonly referred as Pareto's Law (also as the 80:20 rule). Thus, it is very necessary to identify and characterize the 20% of the classes which are change-prone early in the development to enable developers to consider these classes and to redesign them in a proper way.

Moreover, a simple change in a class requires retesting that class, since this change may introduce bugs. Thus, identifying the change-prone classes can enable software testers to focus only on those changed classes.

In the literature, many coupling metrics exist. Some of them were empirically validated against certain software quality attributes such as fault-proneness. Results of the previous empirical studies showed the capability of the coupling metrics in predicting most of the software attributes. So, it is useful to see the capability of coupling metrics to predict the changeability of a software system.

1.3 Research Objective

The objective of the present research is to empirically evaluate and analyze the ability of the investigated coupling metrics to identify the change-prone classes and to predict the density of changes in the classes in the context of evolving object-oriented software systems.

1.4 Research Questions

The research questions of the present thesis are as follows:

- Question 1: Can the investigated coupling metrics identify the change-prone classes and predict the change-density of classes in evolving object-oriented software systems?

- Question 2: Is the model built on coupling metrics more accurate in identifying the change-prone classes and predicting the change-density of classes in evolving object-oriented software systems than the model built on cohesion metrics?

- Question 3: Are the coupling metrics in the C&K suite more accurate than other metrics in the C&K suite in identifying the change-prone classes and predicting the change-density of classes in evolving object-oriented software systems?

- Question 4: Which model is more accurate in identifying the change-prone classes and predicting the change-density of classes in evolving object-oriented software systems: the import coupling metrics model or the export coupling metrics model?

- Question 5: Are all the investigated coupling metrics needed to identify the change-prone classes and predict the change-density of classes in evolving object-oriented software systems?

- Question 6: Is there a confounding effect of class size in the validity of the investigated coupling metrics in identifying the change-prone classes and predicting the change-density of classes in evolving object-oriented software systems?

1.5 Research Contributions

The main contributions of the thesis are summarized as follows:

- Empirical validation of a comprehensive set of coupling metrics to explore their capability to (i) identify change-prone classes; and (ii) predict change-density of classes in a context of evolving object-oriented software systems.

- Construction of several models to identify the change-prone classes and predict change-density of classes, and comparison the accuracy of these models. These models are:
    - Model based on coupling metrics and model based on a subset of coupling metrics,
    - Model based on cohesion metrics and model based on a subset of cohesion metrics,
    - Models based on each metric in C&K suite,
    - Model based on import coupling metrics and model based on a subset of import coupling metrics, and
    - Model based export coupling metrics and model based on a subset of export coupling metrics.
- Investigation of the confounding effect of class size on the validity of the investigated coupling metrics in identifying the change-prone classes and predicting the change-density of classes.

1.6 Thesis Organization

This thesis is organized as follows. Chapter 2 provides information about coupling metrics and frameworks. Reviews of prior empirical studies are presented in chapter 3. Chapter 4 describes the empirical study for exploring the relationship between coupling metrics and change-proneness. Chapter 5 describes the empirical study for exploring

the relationship between coupling metrics and change-density. The conclusion is presented in chapter 6.

Chapter 2

Coupling Metrics

Coupling refers to the degree of interdependence among the modules of a software system. Strong coupling makes a system more complex and hard to understand, change and correct [20]. Several factors constitute coupling, such as inheritance or message passing. Several frameworks of coupling have been proposed in the literature to address the factors which constitute coupling.

2.1 Coupling Measurement Frameworks

Several frameworks have been introduced in the literature to clarify the understanding of the state-of-art of coupling metrics in object-oriented systems. A coupling framework is helpful in providing a standard terminology of coupling, and therefore it can be used to:

1. facilitate comparison of existing metrics,
2. facilitate the evaluation and empirical validation of existing metrics, and
3. support the definition of new metrics and the selection of existing ones based on a particular goal of measurement. [20]

Several authors have proposed frameworks to characterize different approaches which constitute coupling. In an object-oriented system, three frameworks were

proposed for coupling metrics: (1) Framework by Eder et al. [14], (2) Framework by Hitz and Montazeri [24] and (3) Framework by Briand et al. [20]. In each framework, different types of class, method and object coupling are identified.

2.1.1 Framework by Eder et al.

Eder et al. [14] identified three types of relationships in object-oriented systems:

- Interaction relationships between methods,
- Component relationships, and
- Inheritance relationships between classes.

From these relationships, three dimensions of coupling were derived:

1. Interaction Coupling: Two methods are interaction coupled if one method invokes the other, or they communicate via sharing of data.
2. Component Coupling: Two classes c and ć are component coupled, if ć is the type of either an attribute of c, or an input or output parameter of a method of c, or a local variable of a method of c, or an input or output parameter of a method invoked within a method of c.
3. Inheritance Coupling: Two classes *c* and ć are inheritance coupled, if one class is an ancestor of the other.

For each dimension of coupling, Eder et al. identified different strength of coupling. The details of this framework can be found in [14].

2.1.2 Framework by Hitz and Montazeri

Hitz and Montazeri [24] approached coupling by defining the state of an object (the value of its attributes at a given moment at run-time), and the state of an object's implementation (class interface and body at a given time in the development cycle). From these definitions, they derived two levels of coupling:

- Class level coupling (CLC). CLC represents the coupling resulting from state dependencies between two classes in a system during the development lifecycle.
- Object level coupling (OLC). OLC represents the coupling resulting from state dependencies between two objects during the run-time of a system. [24]

According to Hitz and Montazeri, CLC is considered applicable for measuring the maintenance and changeability of a system while OLC is applicable for run-time oriented activities such as testing and debugging. For each level of the coupling, Hitz and Montazeri identified different factors which determine the strength of each level of coupling. [24]

2.1.3 Framework by Briand et al.

The framework of Briand et al. [20] concerns coupling caused by interactions that occur between classes. Interactions between classes are characterized by

- Three Types of interactions:
    - Class-attribute interaction,
    - Class-method interaction, and
    - Method-method interaction.
- Three basic relationships in C++ language

  - Inheritance,
  - Friendship, and
  - Other (No inheritance or friendship).

- Locus. The “locus of impact” of an interaction. If class c is involved in an interaction with another class, a distinction is made between
  - Export. Class c is the used class (server) in the interaction, and
  - Import. Class c is the using class (client) in the interaction.[20]

Briand et al. did not assign strengths to the different kinds of interaction, as previous frameworks had proposed. They stated such strengths should be derived from empirical validation which can then be used to define measures on an interval or ratio scale. Briand et al. pose several hypotheses regarding these facets of coupling and investigate these empirically with respect to prediction of fault-prone classes [20].

2.1.4 Discussion and Comparison of Frameworks

Brief comparison of the frameworks shows that there are differences in the manner in which coupling is considered. This is basically because of the different objectives of each framework and because some of the issues addressed by one author are considered to be subjective by other authors.

Each framework considered some mechanisms that constitute coupling. For example, all frameworks consider a method calling another method as a mechanism of coupling. However, the mechanism of receiving a pointer from a method is considered only by the framework of Briand et al. [20]

Direction of coupling is clearly identified by the framework of Briand et al., unlike the other two frameworks. Briand et al. explicitly distinguished between import and export coupling [20].

Eder et al. derived direct and indirect interaction relationships that constitute coupling. However, other frameworks did not address this issue [14].

Stability of server class is a unique point addressed by Hitz and Montazeri. Stability of a class means that it is unlikely to be changed. Using a stable class is better than using an unstable class because modifications which could ripple through a system are less likely to occur [24].

2.1.5 Selected Framework in the Thesis

In this thesis, the framework by Briand et al. [20] is pursued because of the following reasons:

- It is the only framework that works on a high-level design phase and can investigate potential early quality indicators. [20]
- It covers comprehensively different types of interaction and relation within a class or between classes.
- The issues considered by this framework to determine the kind of interaction can be measured automatically.

However, some metrics in this framework are language-independent. In this thesis, those metrics will not be considered since the aim of this research is to be independent of any programming language so that results can be generalized to other software systems written in any object-oriented programming language.

2.2 Static and Dynamic Coupling Metrics

Coupling metrics can be classified as static and dynamic. Static metrics are those obtained by static analysis of the code i.e., without execution of the code. Dynamic metrics, however, require execution of the code in order to find out its dynamic aspects such as polymorphism. Examples of dynamic coupling metrics can be found in [3]. There are several disadvantages of dynamic coupling. First, they are relatively slow compared to static ones. Second, they require a complete program. Third, the result produced by them is valid only for particular input and execution [36]. In this research, only static coupling metrics are considered.

2.3 Import and Export Coupling Metrics

Briand et al. framework identified two directions of coupling, namely import and export. If a class c is involved in an interaction with another class, a distinction is made between:

- Export: Class c is the used class (server class) in the interaction; and
- Import: Class c is the using class (client class) in the interaction.

This distinction is important. A class which mainly imports services may be difficult to reuse in another context, since it depends on many other classes. In contrast, defects or changes in a class which mainly exports services are particularly critical as they may propagate more easily to other parts of the system and are more difficult to isolate.

2.4 Coupling Metrics Investigated

In this section, the investigated coupling metrics are defined. The coupling metrics used are based on the framework of Briand et al. [20] but coupling metrics which are language-dependent are excluded.

During the definition of each metric, examples will be used to clarify the calculation of each metric. The examples are derived from Figure 1. In the example in Figure 1, there are four classes c1, c2, d1 and d2. The description of each class is as follows:

- Class c1 has two methods: mc1(int a) and mc2(). It has also three attributes of type d1. In both methods, a call is made to method md1() in class d1.
- Class c2 is a child of class c1. It has one attribute of type d1. It has two methods mc2() which is overridden from the parent class and mc3(). In mc2(), a call is made to method md2(int a, int b) of class d1. In mc3(), three methods calls are made one to mc1(int a) of the parent class, one to md1() in class d1 and one to md2(int a, int b) of class d1.
- Class d1 has two methods: md1() and md2(int a, int b).
- Class d2 is a child of class d1. It has two attributes: one of type d1 and one of type c1. It has one method md3() that calls method md1() in the parent class. Class d2 inherits the two methods in the parent class md1() and md2(int a, int b). It also calls method mc2() in class c1.

```
public class c1 {
    d1 *o1;
    d1 *o2;
    d1 *o3;
    public:
           void mc1(int a);
           void mc2();
};
void c1::mc1(int a) {o1 -->md1();}
void c1::mc2() {o1-->md1();}

public class c2: public c1 {

    d1 *o2;
    public:
           void mc2(); /*redefined*/
           void mc3();
};
void c2::mc2() {o2 -->md2(7,1);}
void c2::mc3() {
       mc1(4); o2--> md1();
       o2-->md2(3,1);

       }
```

```
public class d1 {
    public:
           void md1();
           void md2(int a, int b);
};
void d1::md1() {...}
void d1::md2(int a, int b) {...}


class d2: public d1 {
       d1 *o1;
       c1 *o2;
       o2-->mc2()
       public:
              md1();
              md2(2);
              void md3();
};
void d2::md3(){md1()}
```

Figure 1: Example of Coupling Metrics

2.4.1 Coupling Between Objects (CBO)

Chidamber and Kemerer [Chid91][Chid94] developed a suite of metrics for object-oriented systems (the C&K suite) that has heavily influenced most subsequent works. This suite includes two simple coupling metrics: CBO (Coupling Between Objects) and RFC (Response For Class). The original definition of CBO was "CBO for a class is a count of the number of noninhertance related couples with other classes" [31]. There is also a revised definition of CBO that is "CBO for a class is a count of the number of other classes to which it is coupled" [32]. The latter definition of CBO includes coupling due to inheritance while the former did not.

Briand et al. defined two versions of CBO: one based on the first definition and the other based on the second definition (CBO1 and CBO respectively). [20][31][32].

Example

Class c2 is coupled with two classes c1 and d1 since it has attribute o2 of type d1 and it calls mc1(4) methods in class c1. Thus, CBO for class c2 is 2. For CBO1, however, coupling due to inheritance is not counted and thus coupling due to interaction with c1 class is not counted. CBO1 is 1.

2.4.2 Response For Class (RFC)

RFC is one of the metrics in the C&K suite [Chid91][Chid94].RFC is |RS| where RS is the response set for a class. |RS| = $\{M_i\} \cup all_n\{R_i\}$ where $\{M_i\}$ is the set of all methods in class and $\{R_i\}$ is the set of methods called by M_i [Chid94].

This definition indicates that the sets R_i include not only the methods directly invoked by method i, but also the methods called by these methods, and so on. [20]

Briand et al. defined two versions of RFC:

- RFC: The response set of a class consists of the set *M* of methods of the class, and the set of methods directly or indirectly invoked by methods in *M*.
- RFC1: Same as RFC, except that methods indirectly invoked by methods in *M* are not included in the response set. [20][31][32].

Example

Class c2 has two methods mc2() and mc3(). Method mc2() calls method md2(7,1) in class d1. Method mc3 calls three methods mc1() in class c1 and md1() and md2() in class d1. Also, method mc1(4) in class c1 calls method md1(3,1) in class d1.

For RFC, the |RS| of method mc2() is 1 and the |RS| of method mc3() is 4 since the indirect call to method md1() in class c1 is counted. Thus, RFC is 5. For RFC1, however, the indirect call to method md1() in class c1 is not counted and therefore RFC1 is 4.

2.4.3 Message Passing Coupling (MPC)

Li and Henry [34] proposed two coupling metrics, namely MPC (Message Passing Coupling) and DAC (Data Abstraction Coupling). The original definition of MPC is "number of send statements defined in a class". This includes only counting method invocations to other classes, and these classes must not have an inheritance relationship.[34]

Briand et al. argued about the original definition of MPC and they counted inheritance method invocations. They considered invocations of inherited method as send statements. Briand et al. defined MPC as the number of method invocations in a class. They counted only method invocations from other classes, and they included classes with the inheritance relationship [20].

Example

The MPC of class c1 is 2 since it makes two calls to method md1() of class d1.

2.4.4 Data Abstraction Coupling (DAC)

Li and Henry [34] proposed DAC (Data Abstraction Coupling). DAC is defined as the number of abstract data types (ADTs) defined in a class. DAC can also be considered as the number of variables having an ADT type.

Briand et al. [20] provided two versions of DAC:

- DAC: The number of attributes in a class that have another class as their type
- DAC1: The number of different classes that are used as types of attributes in a class.

Example

Class c1 has three abstract data types o1, o2 and o3 of type d1. DAC for class c1 is 3. DAC1 for class c1 is 1 since all abstract data types are of type d1.

2.4.5 Information-flow-based Coupling (ICP)

Lee et al. [37] proposed information-flow based coupling metrics. The original definition of ICP is "ICP(m) counts for method m of class c, the number of polymorphisically invoked methods of other classes, weighted by the number of parameters of the invoked methods".

Briand et al. [20] defined ICP as the number of method invocations in a class, weighted by the number of parameters of the invoked methods (the weight is number of parameters plus 1). Other variations of ICP are IH-ICP and NIH-ICP. IH-ICP is the same as ICP, but it counts only invocations of methods of ancestors of classes (i.e. inheritance-based coupling). NIH-ICP is the same as ICP, but it counts invocations to classes not related through inheritance.

Example

Class c2 makes five method invocations: two calls to md2(int a,int b) in class d1, one call to mc1(int a) in class c1, one call to md1() in class d1 and one call to md1() in method mc1(int a) of class c1. Thus ICP of class c2 is as follows

$$\frac{Total\ Number\ of\ Method\ Invocations}{Total\ Number\ of\ Method\ Invocations + \sum_{i=0}^{all\ methods} parmOf(m_i)}$$

$$\frac{5}{5+\#\ parm.of.md2 + parm.of.mc1 + parm.of.md1 + parm.of.md2 + parm.of.md1}$$

$$= \frac{5}{5+2+1+0+2+0} = 0.5 .$$

For IH-ICP of class c2, only the method invocations to class c1 are considered, and the result will be $\frac{2}{2+1+0} = 0.67$.

For NIH-ICP of class c2, method invocations from inheritance class c1 are not considered, and the result will be $\frac{3}{3+2+0+2} = 0.43$.

2.4.6 Suite of Metrics by Briand et al.

Briand et al. [20] defined a suite of coupling metrics which counts for each class

- The number of class-attribute/class-method/method-method interactions
- Originating from/directed at
- Ancestor/friend/other classes

Briand et al. suite consists of eighteen metrics; six of them are language-dependent. These six rely on relationship occurring because of friendship between classes. Friendship is a relationship in C++ language. When class c declares class d as its friend, it grants class d access to nonpublic elements of class c [20].

Only twelve metrics are considered in this thesis, namely ACAIC, ACMIC, AMMIC, OCAIC, OCMIC, OMMIC, DCAEC, DCMEC, DMMEC, OCAEC, OCMEC and OMMEC. The acronyms for the metrics indicate what interactions are counted:

- the first letter indicates the relationship
    - A – coupling to ancestor classes;
    - D – descendents;
    - O– other.
- the next two letters indicate the type of interaction
    - CA – there is a class-to-attribute interaction between classes C and D, if C has an attribute of type D;
    - CM – there is a class-to-method interaction between classes C and D, if class C has a method with a parameter of type class D;
    - MM – there is a method-to-method interaction between classes C and D, if C invokes a method of D, or if a method of class D is passed as parameter (function pointer) to a method of class C.
- the last two letters indicate the locus of impact
    - IC – import coupling, the measure counts for a class C all interactions where C is using another class;
    - EC – export coupling, counts interactions where class D is the used class.

Example

- Class d2 has one import CA-interaction with its parent class d1 that is o1. Thus, ACAIC(d2) = 1.
- Class d2 has two import CM-interactions with its parent class d1 which are md1() and md2(2). Thus, ACMIC(d2) = 2.

- Class d2 has one import MM-interaction with its parent class d1 that is md1() in method md3(). Thus, AMMIC(d2) = 1.
- Class c1 has three import CA-interactions with class d1 which are o1, o2 and o3. Class c1 and class d1 do not have inheritance relationship. Thus, OCAIC(c1) = 3.
- Class d2 has one import CM-interaction with class c1 that is calling method mc2(). Class c1 and class d2 do not have inheritance relationship. Thus, OCMIC(d2) = 1.
- Class c1 has two import MM-interactions with class d1 which are calling method md1() in both mc1(int a) and mc2(). Class c1 and class d1 do not have inheritance relationship Thus, OMMIC(c1) = 2.
- Class d1 has one export CA-interaction with its descendant class d2 that is o1. Thus, DCAEC(d1) = 1.
- Class d1 has two export CM-interactions with its descendant class d2 which are calling md1() in md2(2) in the parent class. Thus, DCMEC(d1) = 2.
- Class d1 has one export MM-interaction with its descendant class d2 that is calling method md1() of the parent class inside method md3(). Thus, DMMEC(d1) = 1.
- Class d1 has three export CA-interactions with class c1 (o1, o2, o3) and one CA-interaction with class c2 (o2). Both c1 and c2 do not have inheritance relationship with d1. Thus, OCAEC(d1) = 4.
- Class d1 has two export MM-interactions with class c1 (calling md1() in both mc1(int a) and mc2()) and three MM-interactions with class c2 (calling method

md1() inside method mc3() and calling method md2(7,1) inside mc2() and md2(3,1) inside md3()). Both c1 and c2 do not have inheritance relationship with d1. Thus, OMMEC(d1) = 5.

- Class c1 has one export CM-interaction with class d2 that is the called to method mc2() in class d2. Class c1 and d2 do not have inheritance relationship. Thus OCMEC(c1) = 1.

2.5 Discussion of Coupling Metrics Investigated

Coupling metrics investigated might be available at different phases of the software development process. A generic object-oriented development process consists of four phases: analysis, high-level design, low-level design and implementation [20]. Coupling metrics investigated can work at different phases of object-oriented development process.

The type of scale the metrics is defined on is considered as the level of measurement. Levels of measurement are nominal, ordinal, interval and ratio. The empirical relation system used for the attribute is rarely provided. If it is not provided, the indicated scale type reflects by intuitive judgment [20].

Direction of coupling can be either import or export. If a class c is involved in an interaction with another class, a distinction is made between:

- Export: Class c is the used class (server class) in the interaction; and
- Import: Class c is using class (client class) in the interaction.

CBO and CBO1 make no distinction between import and export coupling. Thus they are both considered as import or export coupling metrics [20].

Name	Level of Measurement	Available At	Direction of Coupling
CBO	Ratio	Analysis	Import/Export
CBO1	Ratio	Analysis	Import/Export
RFC	Ordinal	High-level design	Import
RFC1	Ordinal	High-level design	Import
MPC	Ratio	Low-level design	Import
DAC	Ratio	Analysis	Import
DAC	Ratio	Anaysis	Import
ICP	Ratio	Low-level design	Import
NIH-ICP	Ratio	Low-level design	Import
IH-ICP	Ratio	Low-level design	Import
ACAIC	Ratio	Analysis	Import
OCAIC	Ratio	Analysis	Import
DCAEC	Ratio	Analysis	Export
OCAEC	Ratio	Analysis	Export
ACMIC	Ratio	High-level design	Import
OCMIC	Ratio	High-level design	Import
DCMEC	Ratio	High-level design	Export
OCMEC	Ratio	High-level design	Export
AMMIC	Ratio	Low-level design	Import
OMMIC	Ratio	Low-level design	Import
DMMEC	Ratio	Low-level design	Export
OMMEC	Ratio	Low-level design	Export

Table 1: Overview of the coupling metrics investigated

Chapter 3

Literature Review of Prior Empirical Studies

This chapter reviews the prior empirical studies for validating coupling metrics. Coupling metrics were empirically validated against several software quality attributes. Each section below considers a certain software quality attribute for which coupling metrics were validated.

3.1 Empirical Validation of Coupling Metrics with respect to Fault-proneness

Briand et al. [17] conducted an empirical study of the Briand et al. suite (ACAIC, ACMIC, AMMIC, DCAEC, DCMEC, DMMEC, FCAEC, FCMEC, FMMEC, IFCAIC, IFCMIC, IFMMIC, OCAIC, OCMIC, OMMIC, OCAEC, OCMEC and OMMEC). In their empirical study, they intended to test the following hypotheses:

1. The higher the export coupling of a class C, the greater the impact of a change to C on other class. Thus, there is greater likelihood of failure being traced back to faults in C.
2. The higher the import coupling of a class C, the greater the impact of a change in other classes on C itself. Thus, understanding C may be more difficult and therefore more fault-prone, and coupled classes are more likely to be misunderstood and therefore misused by C.

3. Coupling based on friendship between classes in general is likely to increase the likelihood of a fault even more than other types of coupling, since friendship violates modularity in OO design.

The system for their empirical study was in-house development. The system contained fault report forms that described the fault found during testing phase and classes changed to correct such faults. Logistic regression was used to validate coupling metrics. Results showed that:

- OCMEC and OMMEC are good predictors of fault-proneness. This supports the first hypothesis.
- OCAIC, OCMIC and OMMIC metrics are good predictors of fault-proneness. This supports the second hypothesis.
- IFMMIC and FMMEC are good predictors of fault-proneness. This supports the third hypothesis.

Briand et al. [21] conducted also a comprehensive empirical study where they included coupling metrics, cohesion metrics, inheritance metrics and size metrics. The coupling metrics used were the same as those used in the present thesis but they included the language-dependent metrics (FCAEC, FCMEC, FMMEC, IFCAIC, IFCMIC and IFMMIC). The system used in the empirical study is the same as the system used in [21]. Fault-proneness was defined as the probability of detecting a fault in a class. Fault report forms contained the information about faulty classes. In their study, they performed stepwise regression for each group of metrics (namely, coupling metrics, cohesion metrics, inheritance metrics and size metrics) in order to extract from each

group those metrics which are more related to fault-proneness. RFC, RFC1, FMMEC, NIHICP, OCAEC and OMMIC were the resulting metrics of the stepwise regression. The model built on the resulting metrics of the stepwise regression for each group was found to be a good model for predicting fault-proneness.

EL Emam et al. [15] studied ACAIC, OCAIC, DCAEC, OCAEC, ACMIC, OCMIC, DCMEC, OCMEC, AMMIC, OMMIC, DMMEC and OMMEC as well other object-oriented metrics. The system used in their empirical study was commercial Java application. They studied two versions of the systems, and they used failure reports to identify faulty classes. They defined faulty class as a class with at least one fault detected during field operation. They calculated the standard deviation for each coupling metric and they found that only OCAEC, OCAIC, OCMEC, OCMIC met the minimum standard deviation. They calculated also the Spearman correlation of these metrics and they found that OCMEC and OCAEC had the strongest association with fault-proneness.

Gyimóthy et al.[33] studied C&K suite and its two coupling metrics (CBO and RFC). They used an open-source web and e-mail suite called Mozilla. The size of the open source used in their study was large (over a million lines of code). For each metric in the C&K suite they made a hypothesis. The CBO hypothesis was that a class which is more coupled than its peers is more fault-prone. The RFC hypothesis was that a class with larger response-sets than its peers is more fault-prone than they are. Gyimóthy et al. classified a class as faulty if it contains a bug, and bugs of the Mozilla system were available in the database called Bugzilla. They constructed several models based on binary logistic regression, decision-trees and neural networks. Result showed that CBO

and RFC were good predictors of fault-proneness for the model based on binary logistic regression. Also, the decision-trees model and the neural network model confirmed the findings of the regression analysis about CBO and RFC.

Olague et al. [10] studied three object-oriented suites, and one of them was the C&K suite which contains two coupling metrics (CBO and RFC). The system used in their empirical study was called Rhino, and it was developed by using a highly iterative software development process. They studied six versions of Rhino systems, and for each version of Rhino the faulty data was available. In addition, they constructed models based on each object-oriented suite, and the models ware built on binary logistic regression. They did not consider individual metrics inside the C&K suite, but they looked at the suite as a single entity. Result showed that the C&K suite is a good predictor of fault-prone classes.

3.2 Empirical Validation of Coupling Metrics with respect to Testability

Bruntink and Deursen [23] empirically studied the relationship between several object-oriented metrics and software testability. RFC was one of the investigated metrics in their study. Testability was measured in terms of test efforts, taken from the size of test suites. Lines of code (LOC) and number of test cases (NOTC) metrics were used to indicate the size of the test suites. In addition, Bruntink and Deursen conducted the empirical study on five systems. Four of them were commercial systems and the fifth was "Apache Ant". Results showed that there is a significant relationship between RFC and testability.

3.3 Empirical Validation of Coupling Metrics with respect to Reusability

Gui and Scott [7] empirically studied the relationship between CBO, RFC, MPC and DAC and software reusability. Reusability was measured by the number of lines of code that were added, modified or deleted in order to extend some function in the system. Commercial software was used to perform their empirical study, which indicated that there is a strong relationship between reusability and coupling metrics, especially CBO and RFC.

3.4 Empirical Validation of Coupling Metrics with respect to Maintainability

Li and Henry [34] studied coupling metrics (CBO, RFC, MPC, DAC) with respect to maintenance effort. Change effort was measured by the number of lines changed per class. A line change could be an addition or a deletion. Their results showed that there is a strong relationship between coupling metrics and maintenance effort.

3.5 Empirical Validation of Coupling Metrics with respect to Changeability

Wilkie and Kitchenham [9] defined two versions of CBO according to direction of coupling: CBO(back) to count export coupling and CBO(forward) to count import coupling. The system used in their study was a commercial C++ application consisting of two versions. Changeability was measured to show how the final version of the system differs from the first version. Changes were obtained from three sources: (i) bug driven; (ii) customer driven and; (iii) developer driven/redesign. Changes were counted as the number of changes per class. They found that CBO(forward) is good predictor of change-

prone classes. CBO(backward) had slightly lower correlation with change-proneness, and it was found to be not significant.

Koru and Liu studied [1] the Briand et al. suite, cohesion metrics and size metrics. Their aim was to test and validate Pareto's Law which implies that a great majority (around 80%) of changes are rooted in a small proportion (around 20%) of the classes. They also identified and characterized the change-prone classes in two products (KOffice and Mozilla) by producing tree-based models. The changeability was considered as a binary variable (changed or not changed). Their results for both systems strongly supported Pareto's Law. The resulting tree-based model consisted of several metrics, and OCMEC and OCMIC were part of the tree.

Arisholm [4] studied the Briand et al. suite as well as other system-level metrics. Changeability was defined as being capable of change. Commercial java application was used to perform the empirical study. The changeability was considered as a change-density, and it was calculated for each class as follows: the number of added lines of class c, plus the number of deleted lines of class c, divided by the total number of added and deleted lines in all classes. The results showed that the Briand et al. suite can predict change-density, especially those metrics which are related to export coupling [4].

3.6 How this Research Is Different from Previous Research

Several authors in the literature studied coupling metrics with respect to various software quality attributes. Table 2 summarizes of the coupling metrics and the software quality attributes they were validated against. Coupling metrics were previously validated against changeability. This research is different from previous work in terms of the way changeability is considered. This research aims to identify change-prone classes and to predict the change-density of classes. Also, the investigated systems in this research are evolving systems consisting of several releases. Moreover, a comprehensive set of coupling metrics are considered under a unified framework.

Name	Fault-proneness	Testability	Reusability	Maintainability	Changeability
CBO	[21] [10] [21][33]		[7]	[34]	[4][9]
CBO1	[21] [33]				
RFC	[21]	[23]	[7]	[34]	[4]
RFC1	[21]				
MPC	[21]		[7]	[34]	[4]
DAC	[21]		[7]	[34]	[4]
DAC1	[21]				
ICP	[21]				
NIH-ICP	[21]				
IH-ICP	[21]				
ACAIC	[15][17] [21]				[1]
OCAIC	[15][17] [21]				[1]
DCAEC	[15][17] [21]				[1]
OCAEC	[15][17] [21]				[1]
ACMIC	[15][17] [21]				[1]
OCMIC	[15][17] [21]				[1]
DCMEC	[15][17] [21]				[1]
OCMEC	[15][17] [21]				[1]
AMMIC	[15][17] [21]				[1]
OMMI	[15][17] [21]				[1]

C					
DMMEC	[15][17] [21]				[1]
OMMEC	[15][17] [21]				[1]

Table 2: Investigated coupling metrics and software quality attributes they were validated against

Chapter 4

Empirical Study I: Coupling Metrics vs. Change-proneness

In empirical study I, the relationship between coupling metrics and change-proneness of classes in evolving object-oriented software systems will be investigated by using several analyses. For each analysis, observations and results will be discussed.

4.1 Design of Empirical Study I

4.1.1 Goal

The goal of empirical study I is to explore the relationship between coupling metrics investigated and change-proneness of classes in a context of evolving object-oriented software systems. Several models will be built to identify the change-prone classes, and their accuracy will be compared. These models are:

- Model based on coupling metrics and model based on subset of coupling metrics,
- Model based on cohesion metrics and model based on subset of cohesion metrics,
- Models based on each metric in C&K suite,
- Model based on import coupling metrics and model based on subset of import coupling metrics, and

- Model based export coupling metrics and model based on subset of export coupling metrics.

Moreover, the confounding effect of class size in the validity of result obtained by the coupling metrics in identifying the change-prone classes will be investigated.

4.1.2 Motivation

Being able to identify and characterize change-prone classes is very important in different aspects of developing any software system. A simple change in a class requires retesting that class, since this change may introduce bugs. Thus, identifying the change-prone classes can enable program testers to focus only on those classes. Moreover, identifying change-prone classes helps in the process of refactoring a system. Refactoring is usually performed to simplify the structure of a system. Therefore, change-prone classes are candidate classes for the system's designers to look at, because changes which are introduced in these classes may not be clear or consistent with other classes in the same system.

4.1.3 Dependent Variable

The dependent variable is class change-proneness. Change-proneness of a class is defined as a whether class is changed or not from one release to the next release. It is a binary variable for a class, and it can be either changed or not changed.

Changes in a class are considered at line level. If at least one line is changed in a class from one release to the next, this class is considered as a changed class. All white spaces, leading white spaces, trailing white spaces, blank and comments lines are

ignored. The reason of ignoring these features is that this study investigates only the executable lines of code which can introduce bugs in a system and which affect the behavior of a system. Changes in the comments lines, leading spaces or trailing spaces are considered modifications in a system, but treating changes in these lines as changes which occurred in executable lines of code will provide misleading results, since these lines neither affect a system's behavior nor introduce bugs into a system.

4.1.4 Independent Variables

The base metrics used in empirical study I are twenty two coupling metrics (CBO, CBO1, RFC, RFC1, MPC, DAC, DAC1, ICP, IH-ICP, NIH-ICP, OCAIC, OCAEC, OCMIC, OCMEC, OMMIC, OCMMEC, ACAIC, ACMIC, AMMIC, DMMEC, DCAEC and DCMEC) and they can be found in section 2.4. Also, cohesion metrics [19] and the Chidamber and Kemerer (C&K) suite [31][32] are used in empirical study I. The cohesion metrics used in the research are LCOM1, LCOM2, LCOM3, LCOM4, LCOM5, TCC, LCC, ICH, Coh and Co. The metrics in the C&K suite are WMC, LCOM, CBO, RFC, DIT and NOC. The definition of the cohesion metrics and the C&K suite metrics can be found in Appendix A.

4.1.5 Hypotheses

From research questions in section 1.4, we derived the following hypotheses. For each hypothesis, H_0 represents the null hypothesis and H_1 represents its alternative hypothesis.

Hypothesis 1

H_0: There is no association between the investigated coupling metrics and change-prone classes in an evolving object-oriented software system.

H_1: There is an association between the investigated coupling metrics and change-prone classes in an evolving object-oriented software system.

Hypothesis 2

H_0: The accuracy of coupling metrics is lower than or equal to the accuracy of cohesion metrics in identifying the change-prone classes in an evolving object-oriented software system.

H_1: The accuracy of coupling metrics is higher than the accuracy of cohesion metrics in identifying the change-prone classes in an evolving object-oriented software system.

Hypothesis 3

H_0: The accuracy of import coupling metrics is lower than or equal to the accuracy of export coupling metrics in identifying the change-prone classes in an evolving object-oriented software system.

H_1:The accuracy of import coupling metrics is higher than the accuracy of export coupling metrics in identifying the change-prone classes in an evolving object-oriented software system.

Hypothesis 4

H_0: The accuracy of coupling metrics in C&K suite is lower than or equal to the accuracy of other metrics in C&K in identifying the change-prone classes in an evolving object-oriented software system.

H_1: The accuracy of coupling metrics in C&K suite is higher than the accuracy of other metrics in C&K in identifying the change-prone classes in an evolving object-oriented software system.

Hypothesis 5

H_0: All the investigated coupling metrics are needed in identifying the change-prone classes in an evolving object-oriented software system.

H_1: Not all of the investigated coupling metrics are needed in identifying the change-prone classes in an evolving object-oriented software system.

Hypothesis 6

H_0: There is a confounding effect of class size in the validity of the investigated coupling metrics in identifying the change-prone classes in an evolving object-oriented software system.

H_1: There is no confounding effect of class size in the validity of the investigated coupling metrics in identifying the change-prone classes in an evolving object-oriented software system.

4.1.6 Systems Used

Two systems were used in this study. Both are open source systems and written in C++ programming language. Their source codes are available in SourceForge.Net[1]. The first System is named "Stellarium[2]" and the second system is named "LabPlot[3]".

Stellarium is an open-source system started in 18/04/2004. It is considered as an educational system for astronomy, and its goal is to render 3D photo-realistic skies in real time with OpenGL. It displays stars, constellations, planets, nebulas and others things such as ground, landscape, atmosphere, etc. Table 3 shows the number of releases in the system, the date of each release, and the number of classes in each release.

Release Number	Release Date	Number of classes
0.6.2	18/11/2004	102
0.7.1	17/09/2005	117
0.8.0	01/05/2006	140
0.8.1	26/06/2006	147
0.8.2	05/10/2006	154
0.9.0	06/06/2007	191
0.9.1	17/01/2008	190

Table 3: Releases of Stellarium

[1] www.sf.net

[2] www.sourceforge.net/projects/stellarium/

[3] www.sourceforge.net/projects/labplot/

LabPlot is an open-source system started in 18/10/2004. It is considered as a desktop environmental system for visualization data. Its goal is for data plotting and function analysis. Table 4 shows the number of releases in the system, the date of each release, and the number of classes in each release.

Release Number	Release Date	Number of Classes
1.4.0	18/10/2004	35
1.4.1	10/01/2005	38
1.5.0	11/04/2005	44
1.5.1	27/03/2006	49
1.6.0	24/09/2007	53
2.0.0	04/08/2008	24

Table 4: Releases of LabPlot

4.1.7 Data Collection

Two software tools were used to collect independent and dependent variables. These tools are Columbus [4]Framework and Exam Diff Pro[5]. Columbus is a reverse engineering framework that has been developed in cooperation between the University of Szeged, the Nokia Research Center, and FrontEndART. The main motivation for developing the Columbus framework was to create a toolset which supports fact extraction in general and provides a common interface for other reverse engineering tasks as well.

The Columbus framework contains all the necessary components to analyze arbitrary C/C++ source code and to present the extracted information in any desired

[4] www.frontendart.com/

[5] www.prestosoft.com/edp_examdiffpro.asp

form. This framework was used in the present study to collect coupling metrics, cohesion metrics and C&K suite.

ExamDiff Pro is a powerful program for comparing files and directories in the Windows operating system. It offers an efficient and user-friendly way to compare files and folders.

In ExamDiff Pro, several settings were used when comparing files in order to ignore some unnecessary changes for the study. The following changes were ignored in the comparison:

- All white spaces in lines;
- Leading white space in lines;
- Trailing white space in lines;
- Blank lines; and
- Comments.

4.2 Results and Analyses

This section contains the analyses and results performed in empirical study I.

4.2.1 Descriptive Statistics

Table 5 and Table 6 present descriptive statistics for the coupling metrics for Stellarium and LabPlot System respectively.

	MEAN	STD.	MAX	MIN
CBO	2.24	3.91	47	0
CBO1	1.63	3.63	45	0
RFC	16.4	37.49	447	0
RFC1	14.85	34.02	424	0
MPC	16.38	74.08	1309	0
DAC	2.19	12.04	165	0
DAC1	1.09	3.22	34	0
ICP	29.28	141.74	2357	0
IH-ICP	3.13	10.57	109	0
NIH-ICP	26.15	141.51	2357	0
ACAIC	0	0	1	0
DCAEC	0	0	1	0
ACMIC	0	0	2	0
DCMEC	0	0	3	0
AMMIC	0	0	0	0
DMMEC	0	0	0	0
OMMIC	0	0	0	0
OMMEC	0	0	0	0
OCAIC	2.11	11.36	165	0
OCAEC	1.89	5.8	117	0
OCMIC	1.93	4.05	32	0
OCMEC	1.81	7.45	71	0

Table 5: Statistical information of Stellarium

	MEAN	STD.	MAX	MIN
CBO	0.32	0.75	4	0
CBO1	0.31	0.75	4	0
RFC	9.31	17.18	124	0
RFC1	9.31	17.18	124	0
MPC	4.13	15.45	124	0
DAC	0.97	2.21	13	0
DAC1	0.73	1.43	7	0
ICP	4	18.29	168	0
IH-ICP	0	0	0	0
NIH-ICP	4	18.29	168	0
ACAIC	0	0	0	0
DCAEC	0	0	0	0
ACMIC	0	0	0	0
DCMEC	0	0	0	0
AMMIC	0	0	0	0
DMMEC	0	0	0	0
OMMIC	0	0	0	0
OMMEC	0	0	0	0
OCAIC	0.97	2.21	13	0
OCAEC	0.95	2.37	12	0
OCMIC	0.97	2.28	14	0
OCMEC	0.93	2.59	18	0

Table 6: Statistical information of LabPlot

The following are some observations which can be obtained from the Table 5:

- The measures counting coupling of method-method interaction between classes (AMMIC, DMMEC, OMMIC and OMMEC) have less than five non-zero values. This means than there is little use of the method-method interaction between classes in the Stellarium system.

- The high mean and standard deviation of ICP is due to dividing the number of method invocations by the number of parameters of the invoked method.

- The measures counting inheritance coupling between classes (ACAIC, ACMIC, AMMIC, DCAEC, DCMEC, DMMEC and metrics and IH-ICP) have almost zero

values for ACAIC, ACMIC, AMMIC, DCAEC, DCMEC and DMMEC (less than five non-zero values) and relatively low mean and standard deviation for IH-ICP. This means that inheritance was not used a lot in the Stellarium system.

- The method invocations from classes not related through inheritance is high, because of the high mean and standard deviation of NIH-ICP that counts invocation to classes not related through inheritance.

The following are some observations which can be obtained from Table 6:

- The measures counting coupling of method-method interaction between classes (AMMIC, DMMEC, OMMIC and OMMEC) as well as coupling through inheritance (IH-ICP) have less than five non-zero values. This means than there is little use of the method-method interaction between classes and inheritance in the LabPlot system.

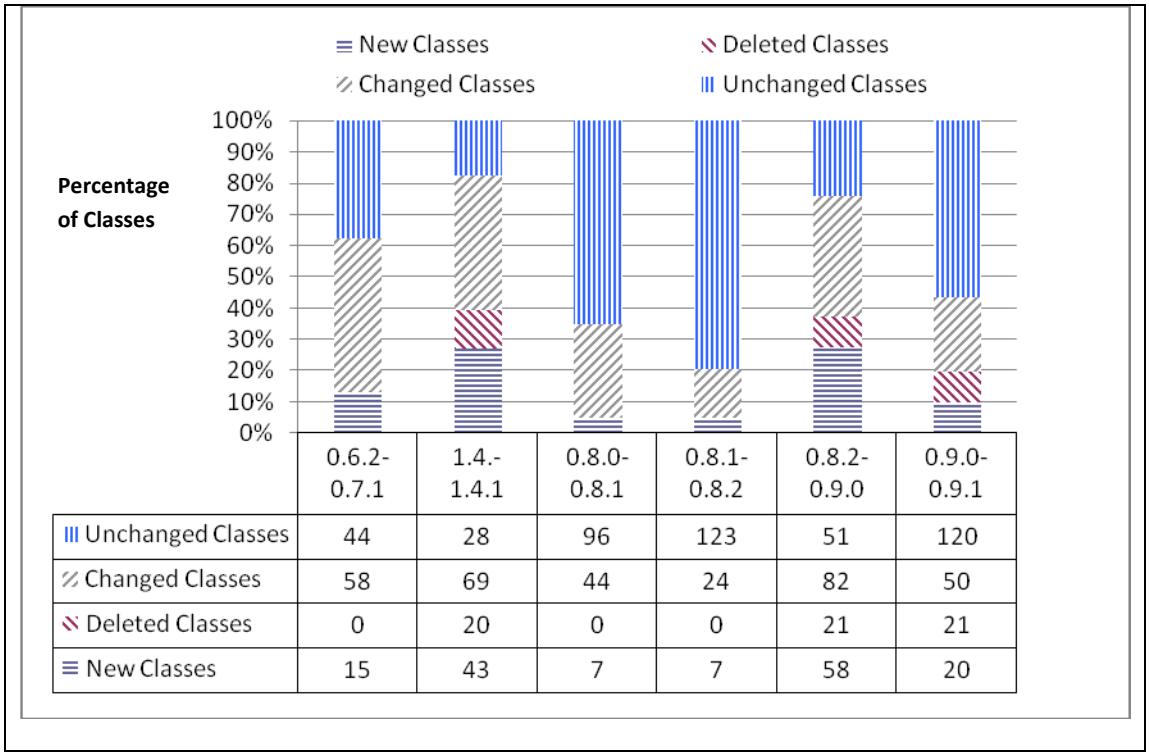
- Both ICP and NIH-ICP have the same values. This means that there is no inheritance in the LabPlot system.

- Both RFC and RFC1 have similar values. This means that the called methods in a class do not call other methods.

In this study , only coupling metrics which have more than five non-zero values will be considered, since coupling metrics with fewer than five non-zero values have low variance (almost zero) and they may mislead the analysis. This method was followed

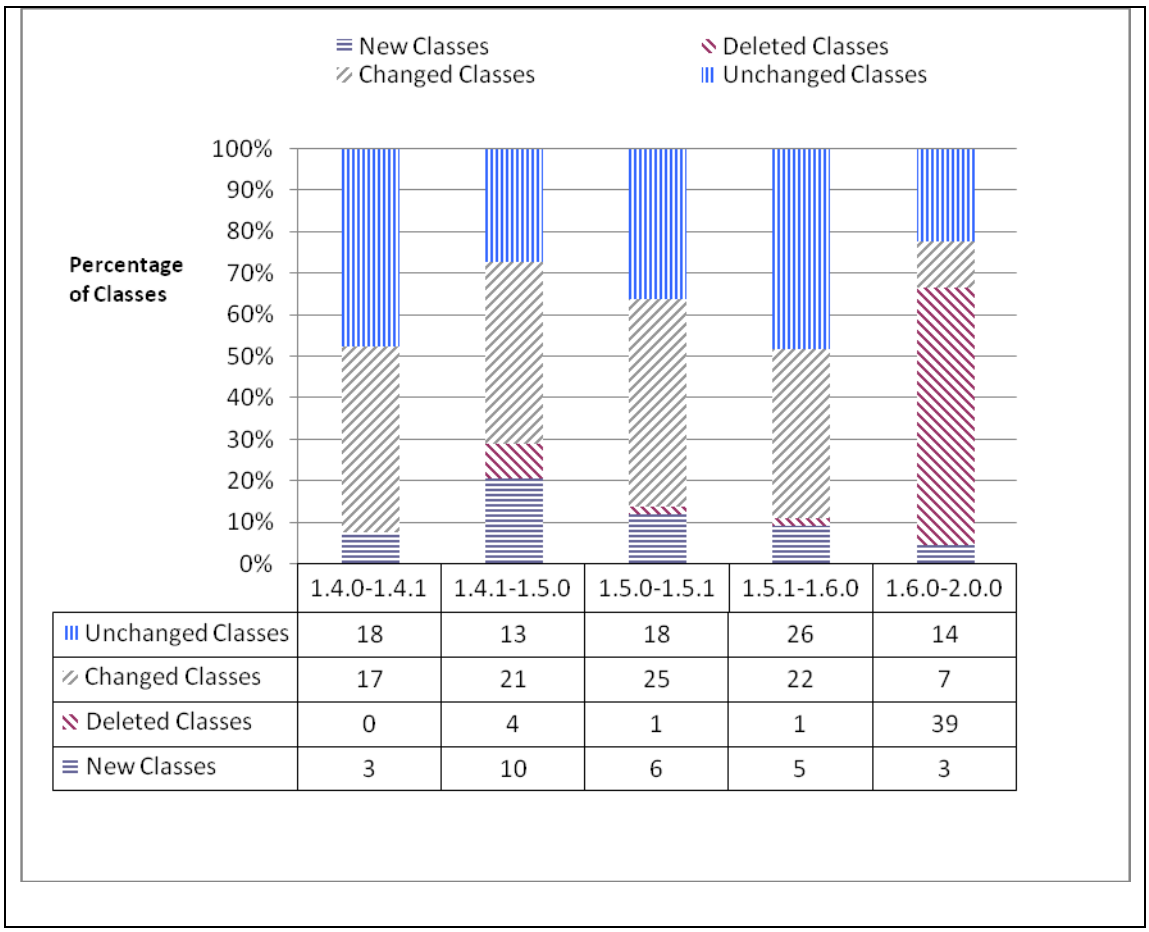
also by Briand et al. [21]. The bolded metrics in Table 5 and Table 6 are the coupling metrics with more than five non-zero values.

The changes which have been performed in both systems are presented in Figure 2 and Figure 3 for Stellarium and LabPlot System respectively.



	0.6.2-0.7.1	1.4.-1.4.1	0.8.0-0.8.1	0.8.1-0.8.2	0.8.2-0.9.0	0.9.0-0.9.1
Unchanged Classes	44	28	96	123	51	120
Changed Classes	58	69	44	24	82	50
Deleted Classes	0	20	0	0	21	21
New Classes	15	43	7	7	58	20

Figure 2: Percentage of changes in Stellarium



	1.4.0-1.4.1	1.4.1-1.5.0	1.5.0-1.5.1	1.5.1-1.6.0	1.6.0-2.0.0
Unchanged Classes	18	13	18	26	14
Changed Classes	17	21	25	22	7
Deleted Classes	0	4	1	1	39
New Classes	3	10	6	5	3

Figure 3: Percentage of changes in LabPlot

It can be observed from the above figures that both systems started with addition of some classes and no classes were deleted. This might be because new features were added to the systems. Both systems were then changed by adding and deleting classes (i.e. from 2nd release to 3rd release). This is normal in any project, since bugs might be discovered and enhancements might be required to improve the clarity and consistency of the systems. After the 3rd release of both systems, it seems that both of them became stable and only some features were only added. This is because of the small number of deleted and added classes between the 3rd and 4th releases and between the 4th and 5th releases. After the 5th release of the LabPlot system, it seems

that there was a complete restructuring of the system, since 73% of the classes were deleted.

After the 5th release of the Stellarium system, it went again through several changes by adding and deleting some classes.

4.2.2 Principle Component Analysis

If a group of variables in a data set are strongly correlated, these variables are likely to measure the same underlying dimension of the object to be measured. Principal Component Analysis (PCA) is a standard technique to identify the underlying orthogonal dimensions that explain relations between the variables in a data set [8].

Principal Components (PCs) are linear combinations of the standardized independent variables. The sum of the squares of the coefficients of the standardized variables in one linear combination is equal to one. PCs are calculated as follows. The first PC is the linear combination of all standardized variables which explain a maximum amount of variance in the data set. The second and subsequent PCs are linear combinations of all standardized variables, where each new PC is orthogonal to all previously calculated PCs and captures a maximum variance under these conditions. Usually, only a subset of all variables have large coefficients, also called the loading of the variable, and therefore contribute significantly to the variance of each PC. The variables with high loadings help identify the dimension the PC is capturing but this usually requires some degree of interpretation [12].

In order to identify these variables, and interpret the PCs, the rotated components are considered. When the PCs are subjected to an orthogonal rotation, the

rotated components show a clearer pattern of loadings, where the variables either have a very low or high loading, thus showing either a negligible or a significant impact on the PC. Several strategies exist to perform such a rotation. Varimax rotation was used, which is the most frequently used strategy in the literature [8].

For a set of n metrics there are, at most, n orthogonal PCs, which are calculated in the decreasing order of variance they explain in the data set. Associated with each PC is its eigenvalue, which is a measure of the variance of the PC. Usually, only a subset of the PCs is selected for further analysis (interpretation, rotated components, etc.). A typical stopping rule, which we also use in this study, is that only PCs whose eigenvalue is larger than 1.0 are selected [12].

Table 7 and Table 8 present the result of PCA of coupling metrics for Stellarium and LabPlot respectively. Values above 0.6 are set in boldface, and these are the coupling metrics called into play when interpreting the PC. The interpretation of each principle component in Stellarium is as follows:

- PC1 (CBO CBO1 RFC RFC1 MPC DAC DAC1 ICP NIH-ICP OCAIC): Measures the extent of import coupling from non-inheritance classes through class-attribute interactions or aggregation.

- PC2 (OCMIC OCMEC): Measures the class-method interaction through import or export coupling.

- PC3 (IH-ICP): Measures the inheritance based coupling from classes.

- PC4 (OCAEC): Measures the extent of export coupling through class-attribute interaction.

The interpretation of each PC in LabPlot System is as follows:

- PC1 (CBO CBO1 RFC RFC1 MPC DAC DAC1 ICP NIH-ICP OCAIC OCMIC): Measures the extent of import coupling from non-inheritance classes through method invocations or aggregation.

- PC2 (OCAEC OCMEC): Measures the extent of export coupling either by class-attribute interaction or class-method interaction.

	Component			
	PC1	PC2	PC3	PC4
CBO	**.917**	.050	.197	.126
CBO1	**.943**	.066	.081	.038
RFC	**.894**	.229	.042	-.021
RFC1	**.873**	.274	.022	-.028
MPC	**.975**	-.056	-.056	-.019
DAC	**.923**	-.208	-.105	-.044
DAC1	**.958**	-.058	-.003	.014
ICP	**.973**	-.124	-.062	.003
IH-ICP	.080	-.037	**.789**	.482
NIH-ICP	**.969**	-.122	-.121	-.033
OCAIC	**.865**	-.188	-.029	-.249
OCAEC	.222	-.024	-.345	**.822**
OCMIC	.207	**.669**	.337	-.210
OCMEC	.030	**.715**	-.404	.171

Table 7: Rotating component with all coupling metrics of Stellarium

	Component	
	PC1	PC2
CBO	**.921**	-.004
CBO1	**.922**	-.003
RFC	**.916**	.074
RFC1	**.916**	.074
MPC	**.954**	.035
DAC	**.941**	-.028
DAC1	**.818**	-.043
ICP	**.922**	.041
NIH-ICP	**.922**	.041
OCAIC	**.941**	-.028
OCAEC	-.090	**.983**
OCMIC	**.869**	.044
OCMEC	-.102	**.981**

Table 8: Rotating component with all coupling metrics of LabPlot

4.2.3 Univariate Logistic Regression

Univariate logistic regression is a standard technique based on maximum likelihood estimation. It is based on the following equation $\pi(x_i) = \frac{e^{(c_0+c_1x_i)}}{1+e^{(c_0+c_1x_i)}}$ where π is the probability that a class contains a change from one release to the next release, and Xi is independent variable which is in this case a coupling metric.

Table 9 and Table 10 summarized the result of univariate analysis for change-proneness for Stellarium and LabPlot System respectively.

Coupling Metrics	R	p-value
CBO	0.14	<0.05
CBO1	0.20	<0.05
RFC	0.19	<0.05
RFC1	0.22	<0.05
MPC	0.20	<0.05
DAC	0.27	<0.05
DAC1	0.27	<0.05
ICP	0.18	<0.05
IH-ICP	-0.16	<0.05
NIH-ICP	0.21	<0.05
OCAIC	0.28	<0.05
OCAEC	0.07	<0.05
OCMIC	0.26	<0.05
OCMEC	0.17	<0.05

Table 9: Univariate analysis of change-proneness for Stellarium

Coupling Metrics	R	p-value
CBO	0.26	<0.05
CBO1	0.24	<0.05
RFC	0.48	<0.05
RFC1	0.48	<0.05
MPC	0.21	<0.05
DAC	0.43	<0.05
DAC1	0.44	<0.05
ICP	0.24	<0.05
NIH-ICP	0.24	<0.05
OCAIC	0.43	<0.05
OCAEC	0.30	<0.05
OCMIC	0.43	<0.05
OCMEC	0.28	<0.05

Table 10: Univariate analysis of change-proneness for LabPlot

For each coupling metric, the Spearman R is provided along with the statistical significance (p-value). Spearman R is the rank correlation coefficient between r pairs of items. It ranges from -1 to 1. The p-value is the probability that the coefficient is

different from zero by chance. The higher the *p-value*, the less it can be believed that the observed relation between variables in the sample is a reliable indicator of the relation between the respective variables in the population. The p-value is set in the analysis to be 0.05.

All coupling metrics have significant relationship with change-proneness of classes in both systems. Also, all coupling metrics in both systems are positively correlated with change-proneness of classes except IH-ICP in Stellarium system.

OCAIC metric has the highest R (0.28) value among all metrics in Stellarium. This shows that there is significant relationship between import coupling through class-attribute interaction and change-proneness of classes. RFC and RFC1 have the highest value of R (0.48) in LabPlot. This indicates that a class with high method invocations is more likely to be change-prone.

In both systems, in general, coupling metrics which count method invocations (e.g. RFC, OCMIC), show a relatively strong value of R. This may indicate that a class with high method invocations is more likely to be change-prone. Also, coupling metrics which count coupling through attribute interaction and aggregation (e.g. OCAIC, DAC) show a significant value of R. This may indicate that a class with many abstract data types defined in it is likely to be more change-prone than other classes. Moreover, coupling metrics which count export coupling (OCAEC, OCMEC) shows relatively low R value. This shows that export coupling does not have a strong relationship with the change-proneness of classes.

From this analysis, the null hypothesis of Hypothesis 1 is rejected and H_1 is accepted that is there is an association between coupling metrics and change-proneness of classes in an evolving object-oriented software system.

4.2.4 Multivariate Logistic Regression

In this section, several models were based on binary logistic regression. The aim of this analysis is to see the capability of the coupling metrics model in predicting the change-proneness of classes.

For each model built in this analysis, the correct classification rate was used to show the accuracy of each model. It is calculated by the following equation:

$$\frac{\#\text{ of classes predicted as changed and they are changed} + \#\text{ of classes predicted as not changed and they are not changed}}{\text{Total number of classes}}$$

A class is classified as change-prone if its predicted probability to contain change is higher than 0.50. The way the analysis conducted was as follows:

- For release i, the model was trained with dataset from release 1 to release i-1
- The model was tested with release i.

The analysis was conducted by using the free software tool called Waikato Environment for Knowledge Analysis (WEKA[6]). The overall goal of WEKA is to build a state-of-the-art facility for developing machine learning (ML) techniques and to apply them to real-world data mining problems. With WEKA, a specialist in a particular field is

[6] www.cs.waikato.ac.nz/ml/weka/

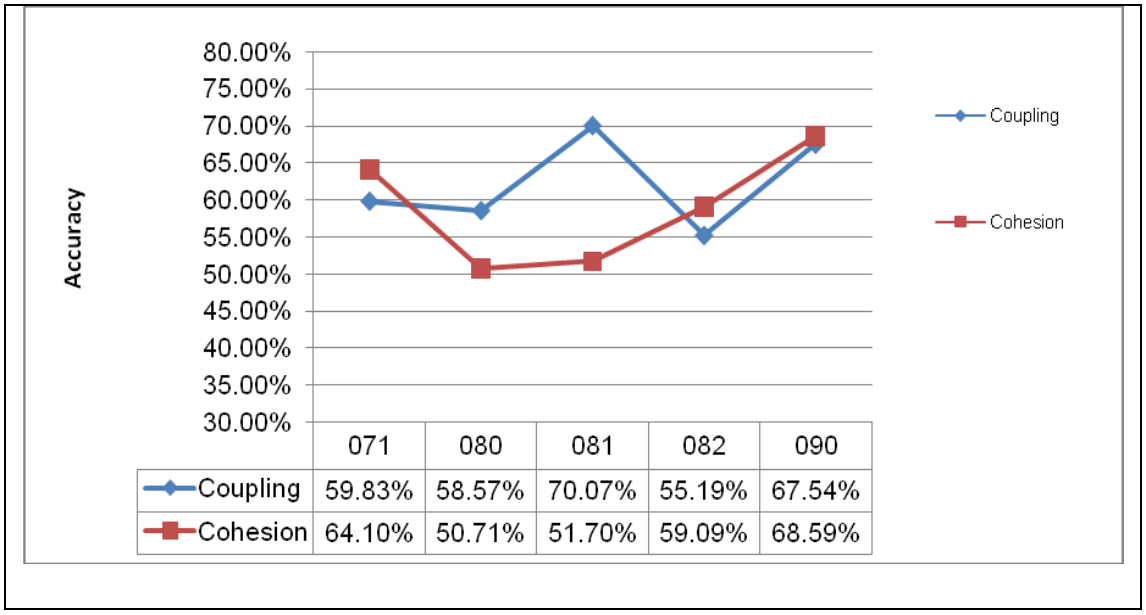
able to use ML to derive useful knowledge from databases that are far too large to be analyzed by hand. WEKA's main users are ML researchers and industrial scientists, but it is also widely used for teaching.

WEKA was used to perform the binary logistic regression for all models created in this analysis. The default settings of WEKA for binary logistic regression were applied. WEKA was also used to select a subset of metrics to be entered in the models for the analysis that required subset selection. The algorithm implemented in WEKA is called "Best-First". Best-first is a machine learning algorithm that is used for attribute selection. It searches the space of attributes for the subset that is most likely to predict the class best. Typically, searching for attributes is greedy in one of two directions, top to bottom or bottom to top. At each stage of searching, a local change is made to the current attribute subset by either adding or deleting a single attribute. The downward direction where you start with no attributes and add them one at a time is called forward selection. In forward selection of best-first, each attribute that is not already in the current subset is tentatively added to it, and the resulting set of attributes is evaluated. [12]

Best-first search is a method that does not just terminate when the performance starts to drop but keeps a list of all attribute subsets evaluated so far, sorted in order of the performance measure, so that it can revisit an earlier configuration instead. Given enough time, it will explore the entire set of attributes, unless this is prevented by some kind of stopping criterion. [12]

4.2.4.1 Coupling Metrics Model vs. Cohesion Metrics Model

Two models were constructed for identifying the change-proneness of classes for each system: (i) one based on coupling metrics; and (ii) one based on cohesion metrics. The accuracy of each model was calculated. Figure 4 and Figure 5 illustrate the accuracy curve of Stellarium and LabPlot respectively. The horizontal axis represents the release number, and the vertical axis represents the accuracy for that release.



	071	080	081	082	090
Coupling	59.83%	58.57%	70.07%	55.19%	67.54%
Cohesion	64.10%	50.71%	51.70%	59.09%	68.59%

Figure 4: Correct classification rate (Accuracy) curves of coupling and cohesion metrics for Stellarium

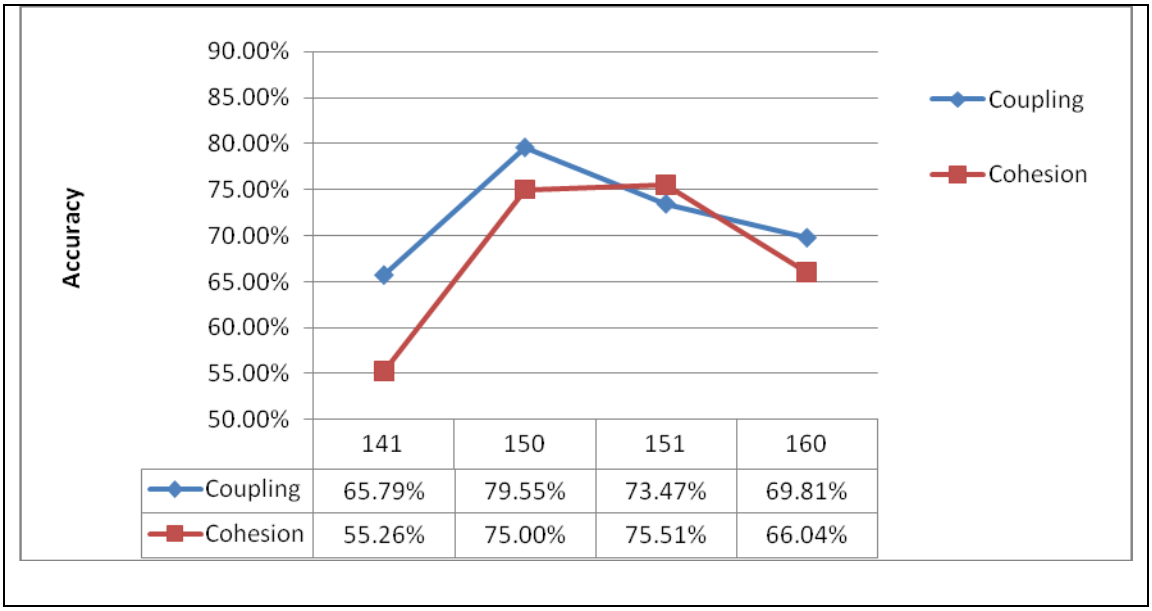


Figure 5: Correct classification rate (Accuracy) curves of coupling and cohesion metrics for LabPlot

In Stellarium, the coupling model had better accuracy in two releases, 080 and 081. In release 081 of Stellarium, the accuracy of the coupling model reached 70.1% which was the best accuracy obtained in all the releases using the coupling model while, the best accuracy in all the releases using the cohesion model reached 68.6% only. The average accuracy of the coupling model was 62% while it was 59% for the cohesion model. Out of five releases of Stellarium, the coupling model was better than the cohesion model in two releases (080 and 081) and the cohesion model was better in two releases as well (071 and 082). In release 090, both models had almost the same accuracy.

In LabPlot, the coupling model had better accuracy in three releases, and in release 151 whose cohesion model was better, the difference was only 2% in the accuracy. The highest accuracy of the coupling model reached 79.6% in release 150 while the highest accuracy in the cohesion model was 75.5%. The average accuracy of

the coupling model was 72% while it was 67% for cohesion model. It can be observed from the results that the models based on coupling metrics outperform the models based on cohesion metrics in both systems.

Out of four releases of LabPlot, the coupling model was better than the cohesion model in three releases (141, 150 and 160) but the cohesion model was better in one release (151).

In general, when considering the average as a way to compare the models, the coupling model was found to be better than the cohesion model. However, when considering the number of times a model is better, both models behaved the same in Stellarium while the coupling model outperformed the cohesion model in LabPlot. As a result, the null hypothesis of Hypothesis 2 is rejected, and the alternative hypothesis is accepted (that is, the accuracy of coupling metrics in identifying the change-prone classes of evolving object-oriented software system is higher than the accuracy of cohesion metrics).

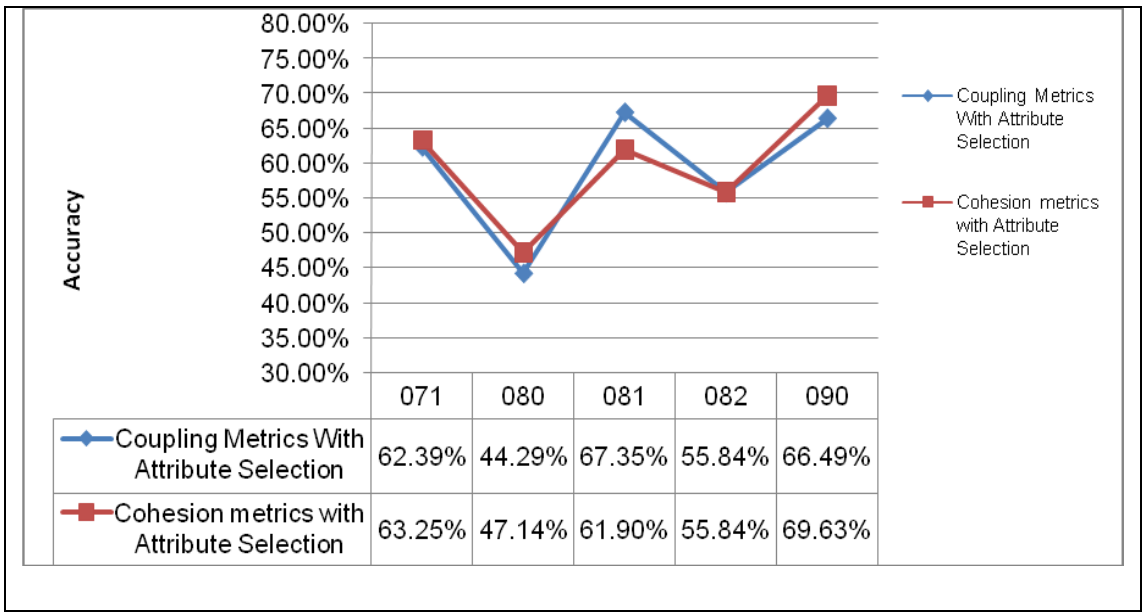
The same analysis was performed again, but this time the pre-processing step was applied to the metrics before they entered the model. The pre-processing step used is an attribute selection technique based on searching called Best-First search. This pre-processing step was used to make the subset selection (refer to Appendix B for more detail).

The resulting subsets of coupling metrics for each release of Stellarium contain CBO1, NIH-ICP and OCMIC along with other metrics. This indicates that these metrics

are working well together. In terms of principle component analysis, CBO1 and NIH-ICP belong to PC1 but OCMIC belongs to PC2. This may show that PC1 and PC2 are the important dimensions of Stellarium. MPC appeared in the subset of the first two releases, which may indicate that this metric is working well in the initial releases of the system. On the other hand, RFC and RFC1 were found in the subsets of the last two releases, which may suggest that both metrics are good in the last releases of the system.

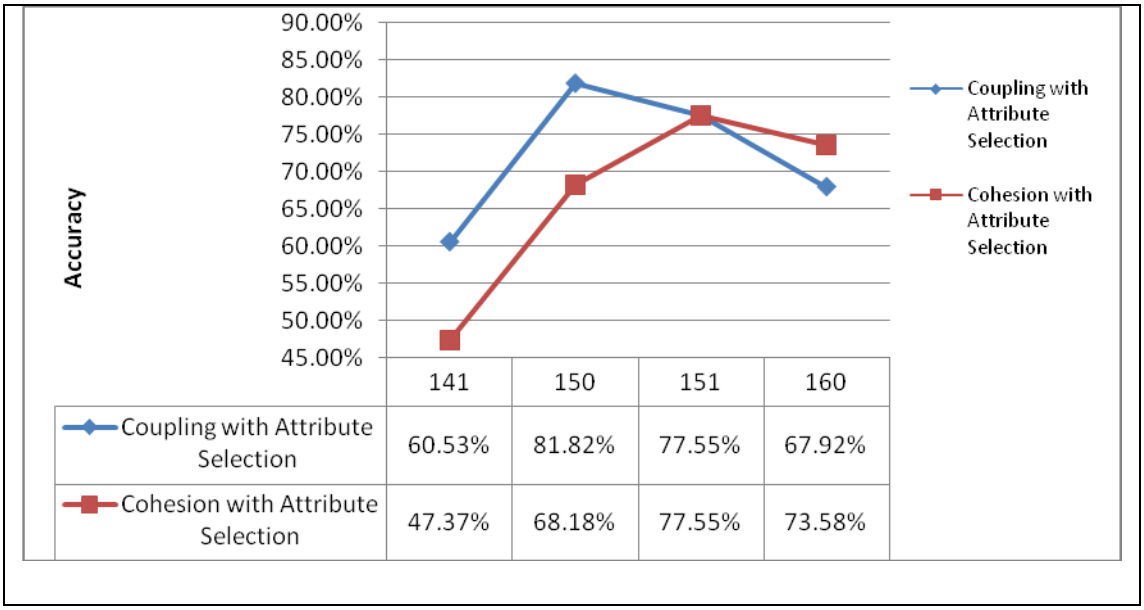
The resulting subsets of coupling metrics for each release of LabPlot contain DAC and OCMIC along with other metrics. This indicates that both metrics are forming a good relationship together. In terms of principle component analysis, both metrics belong to PC1, which indicates that this PC is the dominant dimension of the system.

Figure 6 and Figure 7 illustrate the accuracy curves of Stellarium and LabPlot respectively.



	071	080	081	082	090
Coupling Metrics With Attribute Selection	62.39%	44.29%	67.35%	55.84%	66.49%
Cohesion metrics with Attribute Selection	63.25%	47.14%	61.90%	55.84%	69.63%

Figure 6: Correct classification rate (Accuracy) curves of (subset) coupling and cohesion metrics for Stellarium



	141	150	151	160
Coupling with Attribute Selection	60.53%	81.82%	77.55%	67.92%
Cohesion with Attribute Selection	47.37%	68.18%	77.55%	73.58%

Figure 7: Correct classification rate (Accuracy) curves of (subset) coupling and cohesion metrics for Labplot

In Stellarium, both the coupling model and the cohesion model behaved almost the same. In release 090 of Stellarium, the accuracy of the cohesion model reached 69.6% which was the best accuracy obtained in all the releases using the cohesion model, while the best accuracy in all releases using the coupling model reached 67.35% only. The average accuracy of the coupling and cohesion models was the same (59%).

Out of five releases of Stellarium, the coupling model was better than the cohesion model in one release (081) but the cohesion model was better in two releases (080, 090). In the remaining releases, both models had almost the same accuracy.

In LabPlot, the coupling model had better accuracy in two releases. In release 151, both the coupling and the cohesion models had the same accuracy. For release 160, the cohesion model had better accuracy than the coupling model. The highest accuracy of the coupling model reached 81.8% in release 150, while the highest

accuracy in the cohesion model was 77.5%. The average accuracy of the coupling model was 72% while it was 66% for the cohesion model. It can be observed from the results that the models based on coupling metrics outperform the models based on cohesion metrics in both systems.

Out of four releases of LabPlot, the coupling model was better than the cohesion model in two releases (141 and 150) but the cohesion model was better in one release (160). In release 151, both models had the same accuracy.

When comparing the models based on all metrics and models based on the pre-processing step (refer to Table 11 and Table 12), it can be observed that the pre-processing step improves the accuracy of the cohesion model in Stellarium, while it degrades the accuracy of the coupling model. However, when looking at the difference between the average accuracy of the model based on all coupling metrics and the one with subset selection, only 3% of the accuracy was decreased. In LabPlot, results remained the same in terms of the average accuracy. As a result, the pre-processing step provides an approximately similar result as with all metrics. Thus, the null hypothesis of Hypothesis 5 is rejected, and the alternative hypothesis is accepted (that is, not all of the coupling metrics investigated are needed to identify the change-prone classes in evolving an object-oriented software system).

	Average Accuracy		Number of Times Model Is Better	
	All Metrics	Subset Selection	All Metrics	Subset Selection
Coupling Model	62%	59%	2	1
Cohesion Model	59%	59%	3	3

Table 11: Comparison between models built on all metrics and models built on subset of metrics for Stellarium

	Average Accuracy		Number of Times Model Is Better	
	All Metrics	Subset Selection	All Metrics	Subset Selection
Coupling Model	72%	72%	3	2
Cohesion Model	67%	66%	1	1

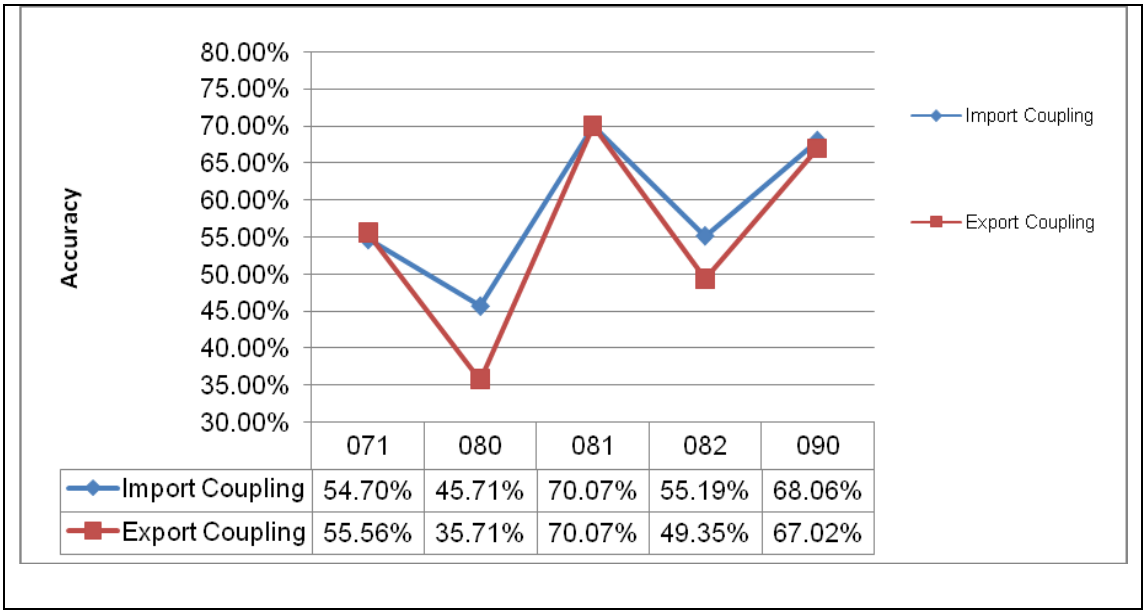
Table 12: Comparison between models built on all metrics and models built on subset of metrics for LabPlot

4.2.4.2 Import Coupling Metrics Model vs. Export Coupling Metrics Model

Coupling metrics can be classified according to import coupling and export coupling, as in section 2.3. Two models are created for identifying the change-proneness of classes: (i) one based on import coupling metrics and; (ii)one based export coupling metrics. Table 13 shows the coupling metrics which are in both the import and the export coupling metrics models. Figure 8 and Figure 9 illustrate the accuracy curves of Stellarium and LabPlot respectively.

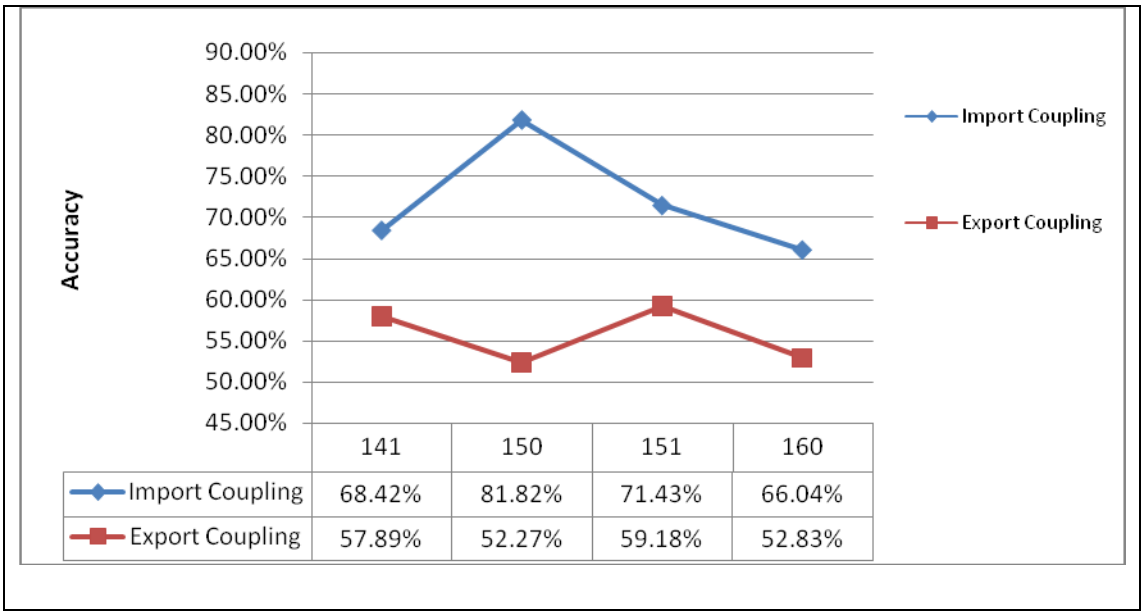
Import Coupling Metrics	CBO CBO1 RFC RFC1 MPC DAC DAC1 ICP IH-ICP NIH-ICP OCAIC OCMIC
Export Coupling Metrics	CBO CBO1 OCAEC OCMEC

Table 13: Import and export coupling metrics



	071	080	081	082	090
Import Coupling	54.70%	45.71%	70.07%	55.19%	68.06%
Export Coupling	55.56%	35.71%	70.07%	49.35%	67.02%

Figure 8: Correct classification rate (Accuracy) curves of import coupling model and export coupling model for Stellarium



	141	150	151	160
Import Coupling	68.42%	81.82%	71.43%	66.04%
Export Coupling	57.89%	52.27%	59.18%	52.83%

Figure 9: Correct classification rate (Accuracy) curves of import coupling model and export coupling model for LabPlot

In Stellarium, the import coupling model had better accuracy in three releases 080, 082 and 090. In release 081, both models had the same accuracy. In release 071, the export coupling model was better than the import coupling model by 1% only. In release 081 of Stellarium, both models had the highest accuracy, which was 70.07%. The

average accuracy of the import coupling model was 58% while it was 55% for the export coupling model.

In LabPlot, the import coupling model outperformed the export coupling model in all releases. The highest accuracy of the import coupling model was 81.8%. The average accuracy of the import coupling model was 71%, which is very high compared with the export coupling model (55%).

As a result of the analysis, the null hypothesis of Hypothesis 3 is rejected, and the alternative hypothesis is accepted (that is, the accuracy of import coupling metrics is higher than the accuracy of export coupling metrics in identifying the change-prone classes of evolving object-oriented software system).

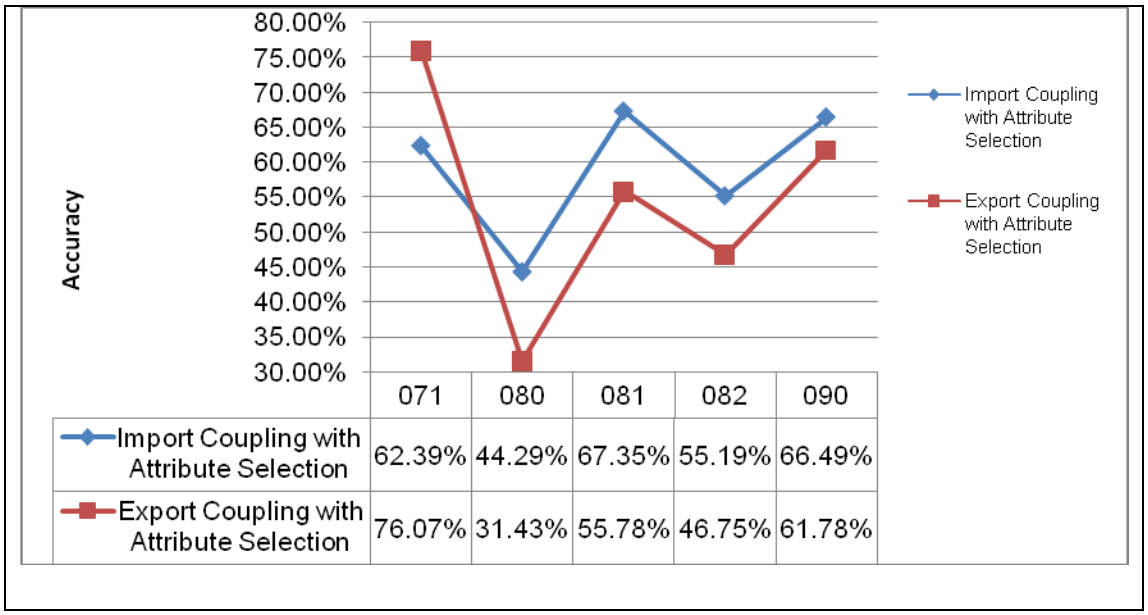
The same analysis was performed again with pre-processing. The resulting subset for each release can be found in Appendix B.

The resulting subsets of import coupling metrics for each release of Stellarium contain NIH-ICP and OCMIC along with other metrics. In terms of principle component analysis, NIH-ICP belongs to PC1 but OCMIC belongs to PC2. This may show that PC1 and PC2 are the important dimensions of Stellarium. MPC appeared in the subset of the first two releases, which may indicate that this metric is working well in the initial releases of the system. On the other hand, RFC was found in the subsets of the last two releases, which may suggest that both metrics are good in the last releases of the system. For resulting subsets of export coupling metrics, CBO1 was found to be in four releases of

Stellarium, which may indicate that this metric is the most important in the export coupling metrics set.

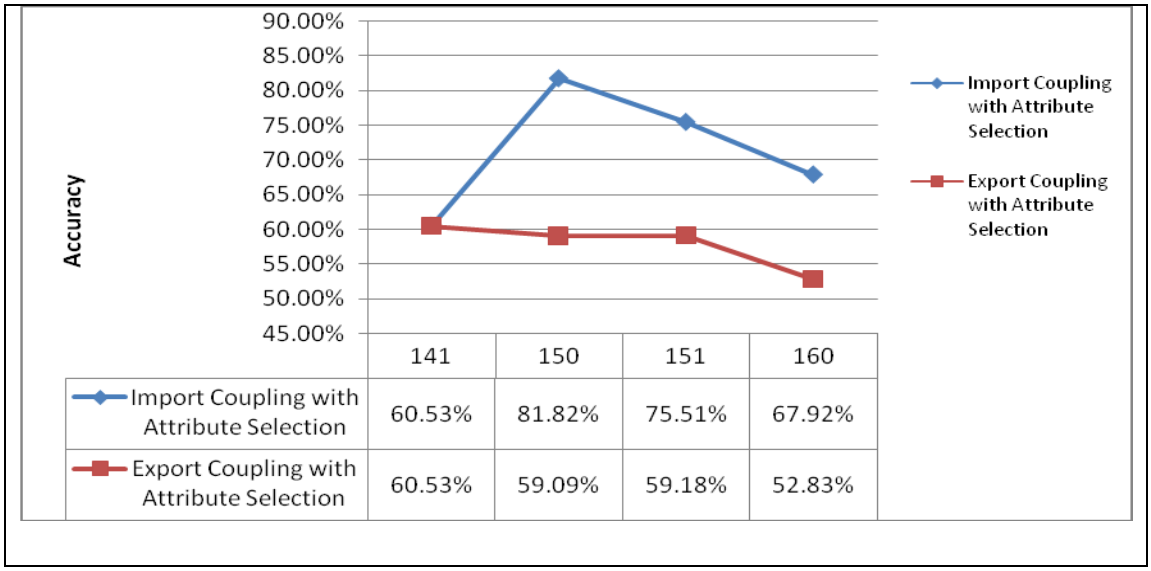
The resulting subsets of coupling metrics for each release of LabPlot contain DAC and OCMIC along with other metrics. Also, RFC was found in the subset of three releases of the system. In terms of principle component analysis, both metrics belong to PC1, which indicates that this PC is the dominant dimension of the system. For export coupling metrics, CBO and OCMEC were found in three releases of LabPlot. This shows that both metrics are working well together.

Figure 10 and Figure 11 illustrate the accuracy curves of Stellarium and LabPlot respectively.



	071	080	081	082	090
Import Coupling with Attribute Selection	62.39%	44.29%	67.35%	55.19%	66.49%
Export Coupling with Attribute Selection	76.07%	31.43%	55.78%	46.75%	61.78%

Figure 10: Correct classification rate (Accuracy) curves of (subset) import coupling model and (subset) export coupling model for Stellarium



	141	150	151	160
Import Coupling with Attribute Selection	60.53%	81.82%	75.51%	67.92%
Export Coupling with Attribute Selection	60.53%	59.09%	59.18%	52.83%

Figure 11: Correct classification rate (Accuracy) curves of (subset) import coupling model and (subset) export coupling model for LabPlot

In Stellarium, the import coupling model had better accuracy in four releases 080, 081, 082 and 090. Surprisingly, the export coupling model had the highest accuracy, which was 76.07% in release 071, while the highest accuracy of the import coupling model was 67.35%. The average accuracy of the import coupling metrics with pre-processing was 59%, while it was only 54% in the export coupling metrics.

In LabPlot, the import coupling metrics with pre-processing had better accuracy in three releases of the system, but both models had the same accuracy in release 141. The average accuracy of the import coupling model with pre-processing is very high compared with the export coupling model with pre-processing (71% to 57%).

When comparing the models with all metrics and the models with pre-processing (refer to Table 14 and Table 15), it can be observed that both models are following the same pattern. For Stellarium, the average accuracy of the import coupling model was 58% while it was 59% with pre-processing. The export coupling model had 55% while the accuracy was 54% with pre-processing. In LabPlot, both the import and

the export coupling models have the same accuracy (71%). The export coupling model with pre-processing had better accuracy than the other one by 2%. As a result, pre-processing seems to provide similar results as with all metrics.

	Average Accuracy		Number of Times Model Is Better	
	All Metrics	Subset Selection	All Metrics	Subset Selection
Import Coupling Model	58%	59%	4	4
Export Coupling Model	55%	54%	1	1

Table 14: Comparison between models built on all metrics and models built on subset of metrics for Stellarium

	Average Accuracy		Number of Times Model Is Better	
	All Metrics	Subset Selection	All Metrics	Subset Selection
Import Coupling Model	71%	71%	4	3
Export Coupling Model	55%	57%	0	0

Table 15: Comparison between models built on all metrics and models built on subset of metrics for LabPlot

4.2.4.3 Models Built Based on Each Metric in C&K Suite

C&K suite consists of six metrics. Two of them are coupling metrics: CBO and RFC. The aim in this analysis is to compare the capability of the coupling metrics in identifying change-proneness of classes with other metrics in the same suite.

For Stellarium, six models were created for identifying the change-proneness of classes for each metric in C&K suite. For LabPlot, models for DIT and NOC were not created, since these two metrics count inheritance but inheritance was not used in LabPlot. Table 16 and Table 17 show the accuracy result for each model for Stellarium and LabPlot respectively. The boldface values indicate the maximum accuracy among other metrics in specified release.

Releases	Accuracy						
	071	080	081	082	090	*AVG.*	*Number of Times Model Is Better*
CBO	**76.07%**	31.43%	60.54%	42.86%	54.45%	*53%*	*1*
RFC	64.10%	31.43%	63.27%	42.86%	59.16%	*52%*	*0*
LCOM	58.12%	31.43%	**72.79%**	40.91%	**67.54%**	***54%***	*2*
WMC	63.25%	**38.57%**	64.63%	**46.75%**	61.26%	***54%***	*2*
NOC	72.65%	32.86%	21.77%	33.12%	62.83%	*44%*	*0*
DIT	53.85%	32.14%	51.70%	33.12%	58.64%	*45%*	*0*

Table 16: Accuracy for each metric in the C&K suite for Stellarium

Releases	Accuracy					
	141	150	151	160	*AVG.*	*Number of Times Model Is Better*
CBO	60.53%	59.09%	57.14%	52.83%	*57%*	*0*
RFC	**63.16%**	**81.82%**	**77.55%**	67.92%	***72%***	*2*
LCOM	47.37%	77.27%	**77.55%**	67.92%	*67%*	*0*
WMC	42.11%	59.09%	75.51%	**69.81%**	*61%*	*1*

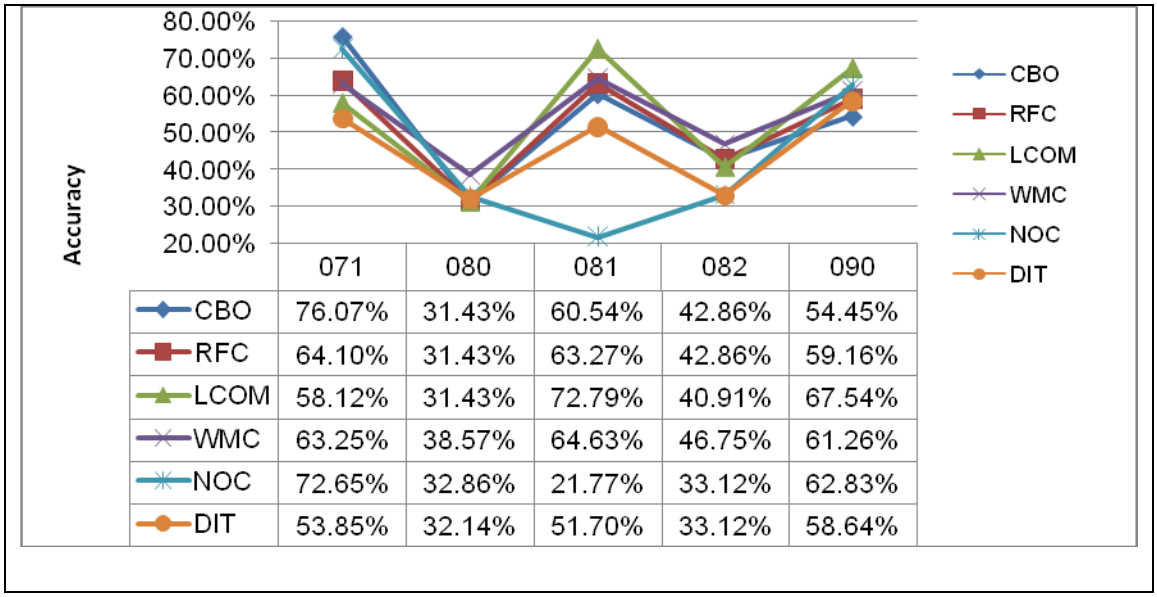
Table 17: Accuracy for each metric in the C&K suite for LabPlot

Coupling metrics in the C&K suite follow a similar pattern as LCOM and WMC in predicting change-prone classes in Stellarium, even though their accuracy was lower than LCOM and WMC. In LabPlot, RFC outperforms all of the metrics in all releases except the last, and the difference between the accuracy of RFC model in that release and the WMC model was small (1.9%). Figure 12 and Figure 13 illustrate the accuracy curves for each model for Stellarium and LabPlot respectively.

In Stellarium, there was reasonable use of inheritance, but the accuracy of NOC and DIT models was not good compared to other models in the C&K suite. This may indicate that inheritance metrics are not good indicators for change-proneness of classes.

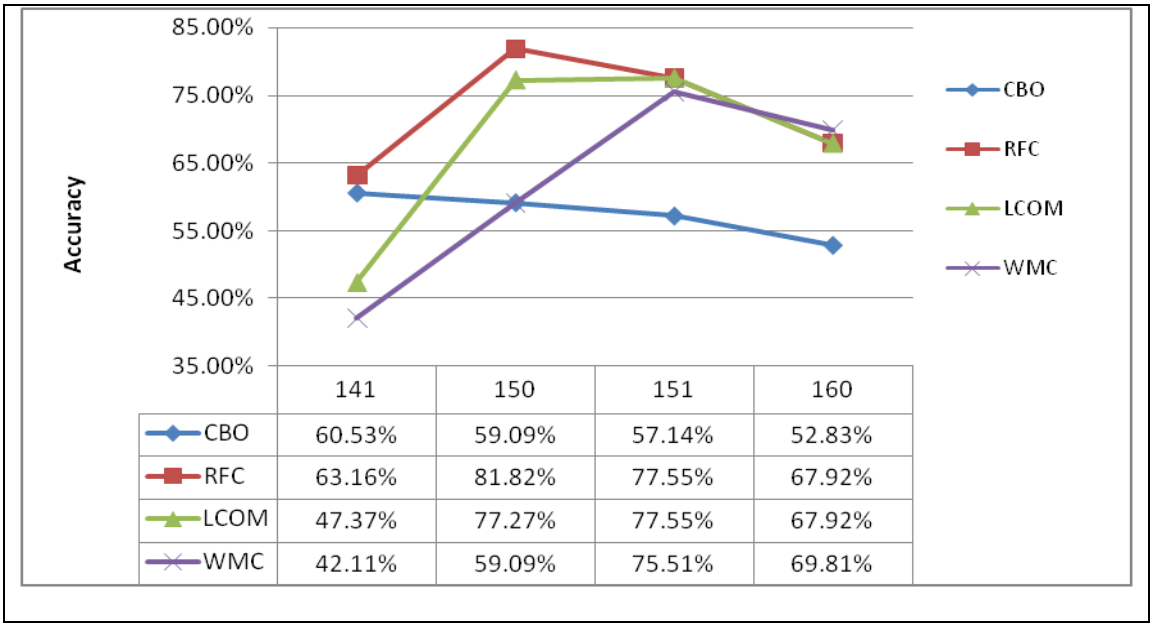
The two systems show different results since in Stellarium, coupling metrics did not outperform other metrics in the C&K suite, while in LabPlot the RFC outperformed

other metrics in all releases. Thus, the null hypothesis is accepted in Stellarium (that is, the accuracy of coupling metrics in the C&K suite is lower than or equal to other metrics in the C&K suite). However, the alternative hypothesis is partially accepted in LabPlot system (that is, the coupling metrics in the C&K suite provide higher accuracy than other metrics in the C&K suite in identifying the change-prone classes in an evolving object-oriented software system). This is because RFC has the highest accuracy in all releases, while the accuracy of CBO was the same or even lower than other metrics in the C&K suite.



	071	080	081	082	090
CBO	76.07%	31.43%	60.54%	42.86%	54.45%
RFC	64.10%	31.43%	63.27%	42.86%	59.16%
LCOM	58.12%	31.43%	72.79%	40.91%	67.54%
WMC	63.25%	38.57%	64.63%	46.75%	61.26%
NOC	72.65%	32.86%	21.77%	33.12%	62.83%
DIT	53.85%	32.14%	51.70%	33.12%	58.64%

Figure 12: Correct classification rate (Accuracy) curves for each model of the metrics in the C&K suite of Stellarium system



	141	150	151	160
CBO	60.53%	59.09%	57.14%	52.83%
RFC	63.16%	81.82%	77.55%	67.92%
LCOM	47.37%	77.27%	77.55%	67.92%
WMC	42.11%	59.09%	75.51%	69.81%

Figure 13: Correct classification rate (Accuracy) curves for each model of the metrics in the C&K suite of LabPlot system

4.2.5 Confounding Effect of Class Size

El Emam et al. [16] studied the confounding effect of class size on the validity of some object-oriented metrics. They included the lines of code in the model along with the object-oriented metrics. They concluded that, if the inclusion of the size to the model has no impact on the result, then there is no significant confounding effect of class size on the validity of the object-oriented metrics.

In section 4.2.3, all metrics were found to have a significant association with the change-proneness of classes. El Emam's procedure will be followed in order to explore whether there is a confounding effect of class size in the result presented in section 4.2.3. In other words, the aim is to explore, after controlling the size confounder, whether all associations between investigated coupling metrics and change-proneness exist.

The model is created twice for each metric in section 4.2.3 to predict change-proneness of classes: (i) one based on individual metric Ci and (ii) based on Ci and size. Size is considered as lines of code.

The Wilcoxon test was used to compare the results obtained from each model. The Wilcoxon test is a nonparametric test that compares two paired groups. It calculates the difference between each set of pairs, and analyzes that list of differences. The P value answers this question: If the median difference in the entire population is zero (the treatment is ineffective), what is the chance that random sampling would result in a median as far from zero (or further) as observed in an experiment?

Table 18 and Table 19 show the results of Stellarium and LabPlot respectively. The bolded values indicate those metrics which are significant at a p-value less than 0.05.

Coupling Metrics	Z	p-value
CBO	**3.288746**	**<0.05**
CBO1	**2.583454**	**<0.05**
RFC	**3.505927**	**<0.05**
RFC1	1.616171	0.106058
MPC	1.611616	0.107046
DAC	1.224721	0.220681
DAC1	1.601534	0.109260
ICP	**3.002917**	**<0.05**
IH-ICP	**5.172904**	**<0.05**
NIH-ICP	1.870777	0.061377
OCAIC	0.394457	0.693244
OCAEC	**4.953772**	**<0.05**
OCMIC	1.300602	0.193396
OCMEC	**4.169322**	**<0.05**

Table 18: Results of the model after controlling the size for Stellarium

Coupling Metrics	Z	p-value
CBO	0.673587	0.500575
CBO1	1.795134	0.072633
RFC	0.534522	0.592980
RFC1	0.534522	0.592980
MPC	0.00	1.000000
DAC	0.00	1.000000
DAC1	0.00	1.000000
ICP	0.00	1.000000
NIH-ICP	0.00	1.000000
OCAIC	0.00	1.000000
OCAEC	0.00	1.000000
OCMIC	1.234130	0.217155
OCMEC	0.00	1.000000

Table 19: Results of the model after controlling the size for LabPlot

The result of RFC1, MPC, DAC, DAC1, NIH-ICP, OCAIC and OCMIC models appear to be insignificant (according to Wilcoxon test) with the result of these models with size. In other words, the result of (RFC1), (MPC), (DAC), (DAC1), (NIH-ICP), (OCAIC) and (OCMIC) models are different from the result of (RFC & LOC) , (MPC & LOC), (DAC & LOC), (DAC1 & LOC), (NIH-ICP & LOC), (OCAIC & LOC) and (OCMIC & LOC) models. This means that the association between these metrics and change-proneness of classes is real regardless of the class's size. For other metrics, after controlling for the size confounder, all the association with change-proneness of classes disappears in Stellarium, while it remains in LabPlot. Thus, no conclusion can be obtained for these metrics.

From this analysis, it can be concluded that the association between some coupling metrics (i.e. coupling metrics with p-value > 0.05) and change-proneness is real. Therefore, the null hypothesis of Hypothesis 6 is partially rejected, and the

alternative hypothesis is partially accepted (that is, there is no confounding effect of class sizes on the validity of some of the investigated coupling metrics.

4.2.6 Threats to Validity

Several issues are identified that affect the results of this empirical study and limit the interpretations. Coupling metrics were found to be good indicators of changeability. However, results were obtained by analyzing classes from only two C++ open source systems. In order to allow for generalization of results, systems from different domains and different programming languages should be studied. In empirical study I also, only static coupling metrics are considered. The results might be somewhat different if dynamic coupling were considered.

Change-proneness of classes is described as a binary variable (changed or not changed). However, the severity of the changes occurred in a class was not considered, and no distinction was made between the changes in the systems. The next chapter considers the change-density of classes from one release to the next. From the change-density, the importance of changes can be characterized according the change-density of class.

In the investigated systems, inheritance was not used much. Thus the effect of the coupling through inheritance, as well as other metrics in cohesion metrics and the C&K suite which count inheritance, was not fully studied.

In addition, this is a regression and correlation study. Association between coupling metrics and change-proneness is confirmed, but causality of the association

cannot be claimed, as altering a class to reduce its coupling metric would decrease its likelihood of being changed. Such claims could only be made after extensive controlled empirical studies into alternative designs.

Chapter 5

Empirical Study II: Coupling Metrics vs. Change-Density

5.1 Design of Empirical Study II

In empirical study II, the relationship between coupling metrics and change-density of classes in evolving object-oriented software systems will be investigated by using several analyses. For each analysis, observations and results will be discussed.

Several issues of the design of empirical study II can be found in section 4.1. The independent variables are those in section 4.1.4. The systems used in empirical study II, and the data-collection procedure, can be found in section 4.1.6 and 4.1.7 respectively.

5.1.1 Goal

The goal of empirical study II is to explore the relationship between coupling metrics investigated and change-density of classes in a context of evolving object-oriented software systems. Several models will be built to predict the change-density of classes, and their accuracy will be compared. These models are:

- Model based on coupling metrics and model based on subset of coupling metrics,
- Model based on cohesion metrics and model based on subset of cohesion metrics,
- Models based on each metric in C&K suite,

- Model based on import coupling metrics and model based on subset of import coupling metrics, and
- Model based export coupling metrics and model based on subset of export coupling metrics.

Moreover, the confounding effect of class size on the validity of results obtained by the coupling metrics in identifying the change-prone classes will be investigated.

5.1.2 Motivation

Being able to predict the density of change is very important in different aspects of software development. A class which has a large amount of changes requires more time and care when testing it, since changes to that class may introduce bugs or may be inconsistent with other parts of the system. Also, having the knowledge of the density of changes which happened to a system allows the designer or tester of a system to distribute time and effort in the project more accurately, since a high amount of changes in a system requires usually more time and effort in testing and refactoring.

5.1.3 Dependent Variable

The dependent variable of empirical study II is change-density. Change-density is defined as the amount of changes in a class from one release to the next, normalized by the lines of code of that class. It is calculated by the formula [34]:

$$\frac{Number\ of\ added\ lines + number\ of\ deleted\ lines + (2 \times number\ of\ changed\ lines)}{lines\ of\ code}$$

The reason for multiplying the number of changed lines by two is that the changed lines require addition and deletion. [34]

Changes in a class are considered at line level. All white spaces, leading white spaces, trailing white spaces, blank and comments lines are ignored. The reason for ignoring these features is that the present empirical study investigates only the executable lines of code which affect the behavior of a system.

5.1.4 Hypotheses

From research questions in section 1.4, we derived the following hypotheses. For each hypothesis, H_0 represents the null and H_1 represents the negation of the null.

Hypothesis 1

H_0 : There is no association between the investigated coupling metrics and the change-density of classes in an evolving object-oriented software system.

H_1 : There is an association between the investigated coupling metrics and the change-density of classes in an evolving object-oriented software system.

Hypothesis 2

H_0 : The accuracy of coupling metrics is lower than or equal to the accuracy of cohesion metrics in predicting the change-density of classes in an evolving object-oriented software system.

H_1 : The accuracy of coupling metrics is higher than the accuracy of cohesion metrics in predicting the change-density of classes in an evolving object-oriented software system.

Hypothesis 3

H_0 : The accuracy of import coupling metrics is lower than or equal to the accuracy of export coupling metrics in predicting the change-density of classes in an evolving object-oriented software system.

H_1 :The accuracy of import coupling metrics is higher than the accuracy of export coupling metrics in predicting the change-density of classes in an evolving object-oriented software system.

Hypothesis 4

H_0 : The accuracy of coupling metrics in C&K suite is lower than or equal to the accuracy of other metrics in the C&K suite in predicting the change-density of classes in an evolving object-oriented software system.

H_1 :The accuracy of coupling metrics in the C&K suite is higher than the accuracy of other metrics in C&K suite in predicting the change-density of classes in an evolving object-oriented software system.

Hypothesis 5

H_0 :All the investigated coupling metrics are needed in predicting the change-density of classes in an evolving object-oriented software system.

H_1 : Not all of the investigated coupling metrics are needed in predicting the change-density of classes in an evolving object-oriented software system.

Hypothesis 6

H_0: There is a confounding effect of class size on the validity of the investigated coupling metrics in predicting the change-density of classes in an evolving object-oriented software system.

H_1: There is no confounding effect of class size on the validity of the investigated coupling metrics in predicting the change-density of classes in an evolving object-oriented software system.

5.2 Results and Analyses

In this section, analyses and results of the created model for predicting the change-density of classes are discussed.

5.2.1 Univariate Regression

Table 20 and Table 21 summarized the results of univariate analysis for change-density of classes for Stellarium and LabPlot respectively. Univariate analysis was performed in the classes which were changed, in order to predict the change-density of classes and because unchanged classes have zero values for the change-density. Thus, the inclusion of unchanged classes may provide misleading results.

Coupling Metrics	R	p-value
CBO	**-0.222**	**<0.05**
CBO1	**-0.216**	**<0.05**
RFC	**-0.248**	**<0.05**
RFC1	**-0.251**	**<0.05**
MPC	**-0.262**	**<0.05**
DAC	**-0.161**	**<0.05**
DAC1	**-0.138**	**<0.05**
ICP	**-0.237**	**<0.05**
IH-ICP	-0.019	0.707353
NIH-ICP	**-0.235**	**<0.05**
OCAIC	**-0.179**	**<0.05**
OCAEC	**-0.106**	**<0.05**
OCMIC	**-0.124**	**<0.05**
OCMEC	-0.092	0.067178

Table 20: Univariate analysis of change-density for Stellarium

Coupling Metrics	R	p-value
CBO	-0.172	0.056559
CBO1	-0.152	0.092946
RFC	-0.091	0.313125
RFC1	-0.091	0.313125
MPC	-0.122	0.178485
DAC	**-0.179**	**<0.05**
DAC1	-0.167	0.063886
ICP	-0.133	0.141351
NIH-ICP	-0.133	0.141351
OCAIC	**-0.179**	**<0.05**
OCAEC	-0.083	0.357624
OCMIC	-0.027	0.764791
OCMEC	0.020	0.820955

Table 21: Univariate analysis of change-density for LabPlot

For each coupling metrics, the Spearman R is provided along with the statistical significance (p-value). The bolded values indicate the coupling metrics in which p-value <0.05.

In both systems, all metrics which show a significant association with the change-density have a negative R value. Looking at the density formula, the amount of changes

in a class is weighted by the lines of code. In most cases, the density is less than one (i.e. most of the classes do not have changes greater than lines of code). Thus, a negative correlation between coupling metrics and change-density of classes is normal, because of the weight of lines of code.

MPC metric has the highest R value (-0.26) in Stellarium. This shows that there is a significant relationship between coupling through method invocations and change-density. When considering the p-value of DAC, CBO, CBO1 and DAC1, the p-value was less than 0.1, and these metrics are the only metrics significant in LabPlot. This may indicate that there is a significant relationship between coupling through aggregations and change-density of classes.

In both systems in general, export coupling through class method interaction (OCMEC) does not have a strong relationship with change-density, even though this metric has more than five non-zero values. However, it has a low mean and standard deviation compared with other metrics in the system. This may explain why it failed to have significant association with the change-density of classes.

From this analysis, the null hypothesis of Hypothesis 1 is partially rejected, and the alternative hypothesis is partially accepted (that is, there is an association between some coupling metrics and the change-density of classes in an evolving object oriented software system).

5.2.2 Multivariate Regression

In this section, several models were based on linear regression. The aim of this analysis is to see the capability of the coupling metrics model in predicting the change-density of classes in an evolving object-oriented software system. Models were built in this study with classes which were changed, in order to predict the change-density of classes and because unchanged classes have zero values for the change-density. Thus, the inclusion of unchanged classes may provide misleading results.

For each model built in this analysis, Mean Magnitude of Relative Error (MMRE) and PRED(0.25) are used for accuracy of estimation, and they are considered as the correct classification rate. MMRE is based on the following formula:

$$MMRE = \frac{1}{N} \times \sum_{i=1}^{N} \frac{|Actual_i - Estimated_i|}{Actual_i} \; Where$$

$$Actual_i \, is \, the \, actual \, change \, density \, of \, class \, i \, and;$$

$$Estimated_i \, is \, the \, estimated \, change \, density \, of \, class \, i$$ [12]

PRED(0.25) can be calculated by the following formula:

$$PRED(0.25) = \frac{n}{N} \; where$$

$$n = Number \, of \, times \, \frac{|Actual_i - Estimated_i|}{Actual_i} \leq 0.25 \, and \, N \, is \, the \, number \, of \, classes;$$

The way the analysis conducted was similar to empirical study I, that is:

- For release i, the model was trained with dataset from release 1 to release i-1
- The model was tested with release i.

WEKA was used to perform the linear regression for all models created in this analysis as well as subset selection as in empirical study I (refer to section 4.2.4 for more details). The default settings for linear regression were used, except that no attribute selection method was used and the collinear attribute was not removed in the analysis. The following changes were made to the settings in order to allow all metrics to enter the model, since these settings may prevent some metrics from entering the constructed models.

5.2.2.1 Coupling Metrics Model vs. Cohesion Metrics Model

Two models were constructed for each system for predicting the change-density of classes: (i) one based on coupling metrics; and (ii) one based on cohesion metrics. The MMRE values for each release of the created models are presented in Figure 14 and Figure 15 for Stellarium and LabPlot respectively. For PRED(0.25) values, refer to Appendix C for more details.

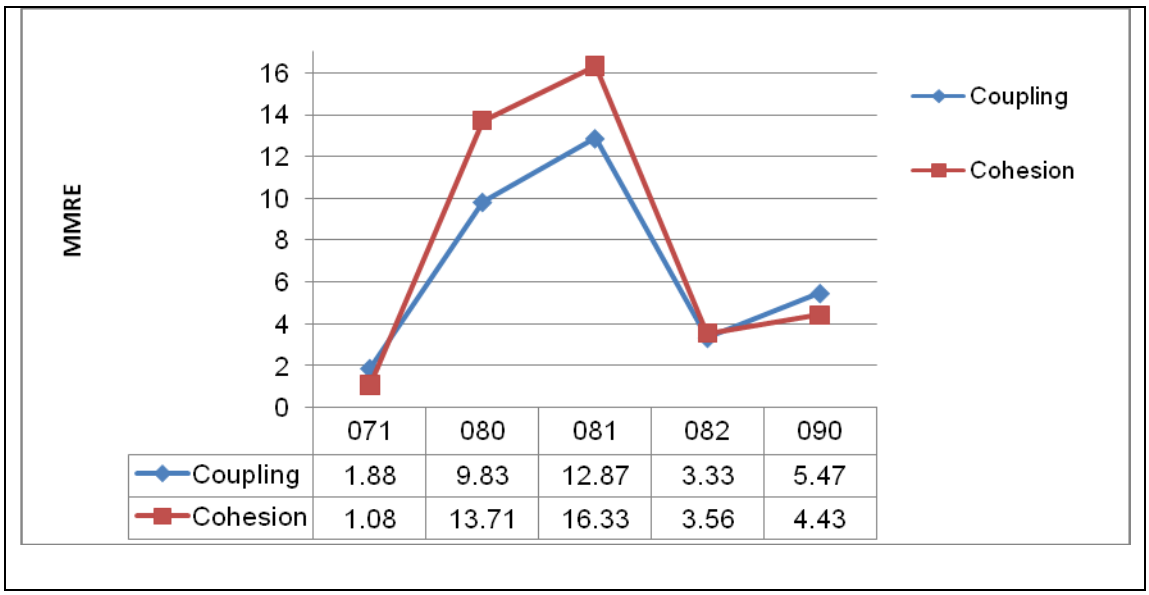


Figure 14: MMRE curves for coupling and cohesion metrics model for Stellarium

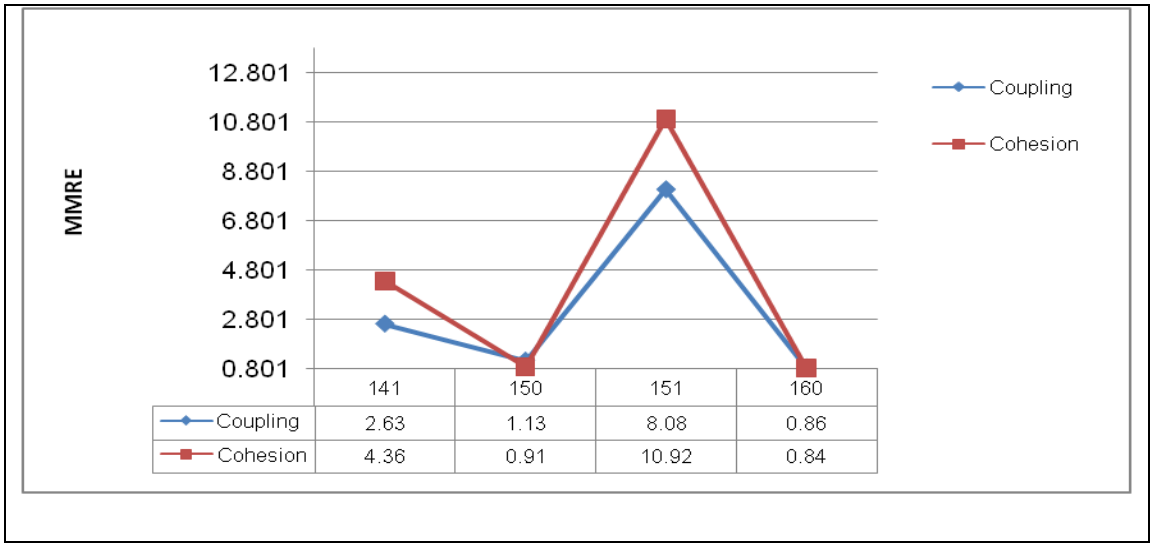


Figure 15: MMRE curves for coupling and cohesion metrics model for LabPlot

In Stellarium, the coupling metrics model had a lower MMRE value in three releases (080, 081 and 082). In terms of the average MMRE for all releases, the coupling metrics model had 6.68 while it was 7.82 for the cohesion metrics model. In general for Stellarium, the coupling metrics model outperforms the cohesion metrics model.

In two releases of LabPlot, the coupling metrics model outperformed the cohesion metrics model but the opposite occurred in one release. The average MMRE

values for the coupling metrics model was 3.18 whereas it was 4.26 for the cohesion metrics model.

As a result, the null hypothesis of Hypothesis 2 is rejected, and the alternative hypothesis is accepted (that is, the accuracy of coupling metrics is higher than the accuracy of the cohesion metrics model in predicting the change-density of classes in an evolving object-oriented software system).

The same analysis was performed again, but this time the pre-processing step was applied to the metrics before they entered the model. The pre-processing step is an attribute selection technique based on searching, called Best-First search. For more information about the attribute selection technique, refer to section 4.2.4. The resulting subset for each release can be found in Appendix B.

Figure 16 and Figure 17 illustrate the result for the models built on attribute selection for Stellarium and LabPlot respectively.

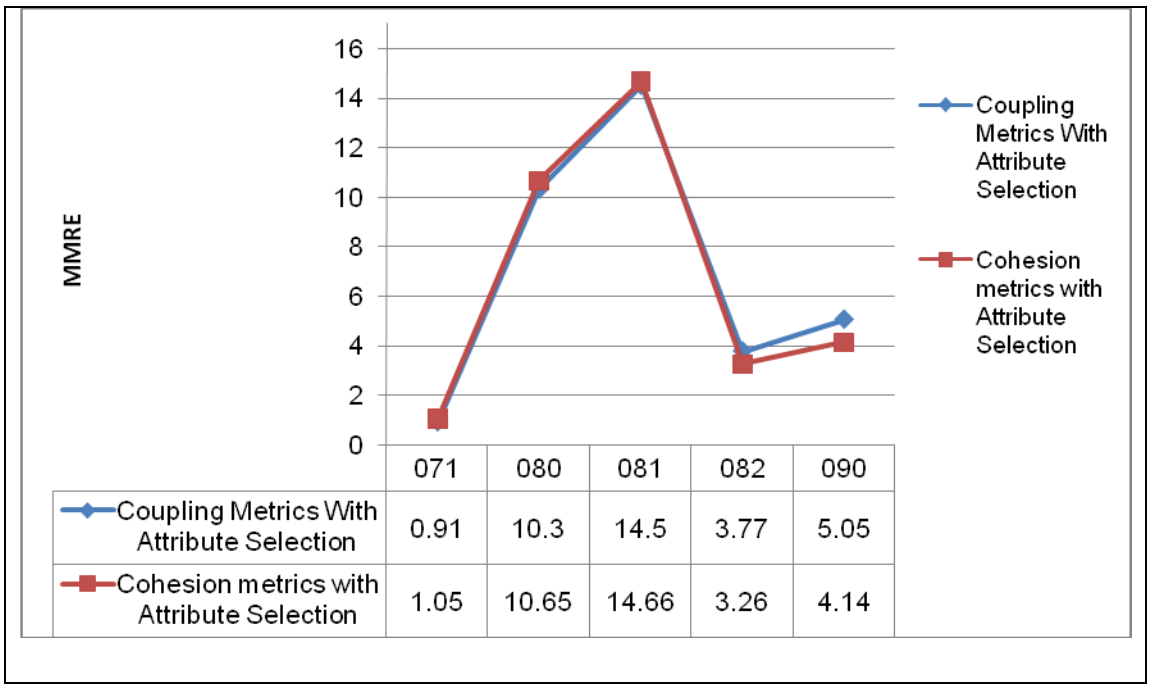


Figure 16: MMRE curves for (subset) coupling and (subset) cohesion metrics model for Stellarium

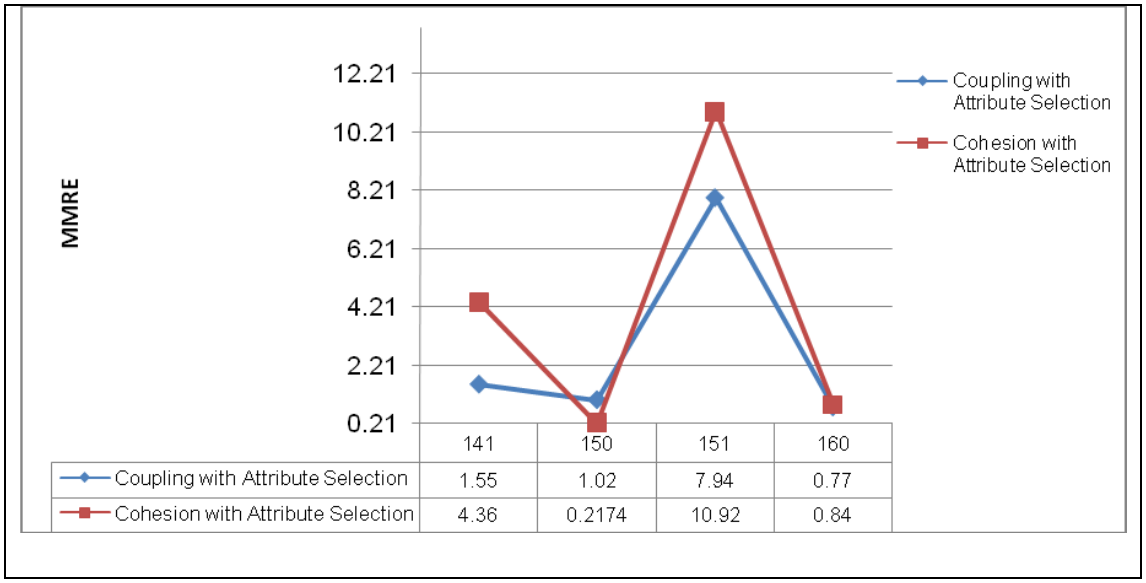


Figure 17: MMRE curves for (subset) coupling and (subset) cohesion metrics model for LabPlot

In Stellarium, both models performed almost the same in terms of MMRE values. Moreover, Appendix C indicates that both models performed almost the same in terms of the PRED(0.25) value. The model based on a subset of coupling metrics was slightly better in releases 081, 080 and 081, but the model based on a subset of cohesion metrics was better in the remaining releases.

In LabPlot, the MMRE results show clearly that the coupling metrics model outperforms the cohesion metrics model. The coupling model had better accuracy in three releases (141, 151 and 160). The average MMRE values for the coupling metrics model is 2.82 while it is 4.08 for the cohesion metrics model.

When comparing the models based on all metrics and models based on the pre-processing step (refer to Table 22 and Table 23), it can be observed that the number of times a model was better using the subset was similar to the number for the model based on all metrics in in Stellarium (3). In LabPlot, the number of times a model was better using the subset were greater than the model based on all metrics by one. However, the average MMRE values of the models based on pre-processing for coupling metrics and cohesion metrics were slightly different. In general, the pre-processing step seems to provide almost similar results as with the model created with all metrics. The null hypothesis of Hypothesis 5 is rejected, and the alternative hypothesis is accepted (that is, not all of the investigated coupling metrics are needed to predict the change-density of classes in evolving object-oriented software systems).

	Average MMRE		Number of Times Model Is Better	
	All Metrics	Subset Selection	All Metrics	Subset Selection
Coupling Model	6.68	6.91	3	3
Cohesion Model	7.82	6.75	2	2

Table 22: Comparison between models built on all metrics and models built on subset of metrics for Stellarium

	Average MMRE		Number of Times Model Is Better	
	All Metrics	Subset Selection	All Metrics	Subset Selection
Coupling Model	3.18	2.82	2	3
Cohesion Model	4.26	4.08	2	1

Table 23: Comparison between models built on all metrics and models built on subset of metrics for LabPlot

5.2.2.2 Import Coupling Metrics Model vs. Export Coupling Metrics Model

Two models were created in this analysis for predicting the change-density of classes, (i) one based on import coupling metrics and (ii) one based on export coupling metrics. The aim of this analysis is to investigate which model is more accurate. For more information about import coupling metrics and export coupling metrics, refer to section 2.3 and section 4.2.4.2.

Figure 18 and Figure 19 illustrate the MMRE value for each release for Stellarium and LabPlot respectively.

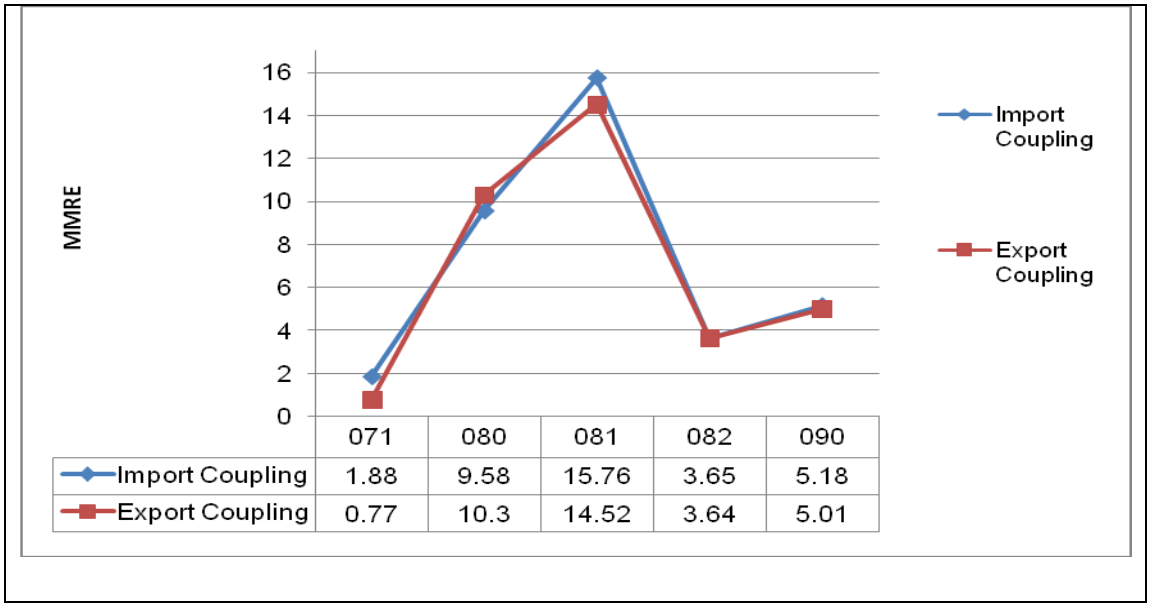


Figure 18: MMRE curves for import coupling metrics and export coupling metrics model for Stellarium

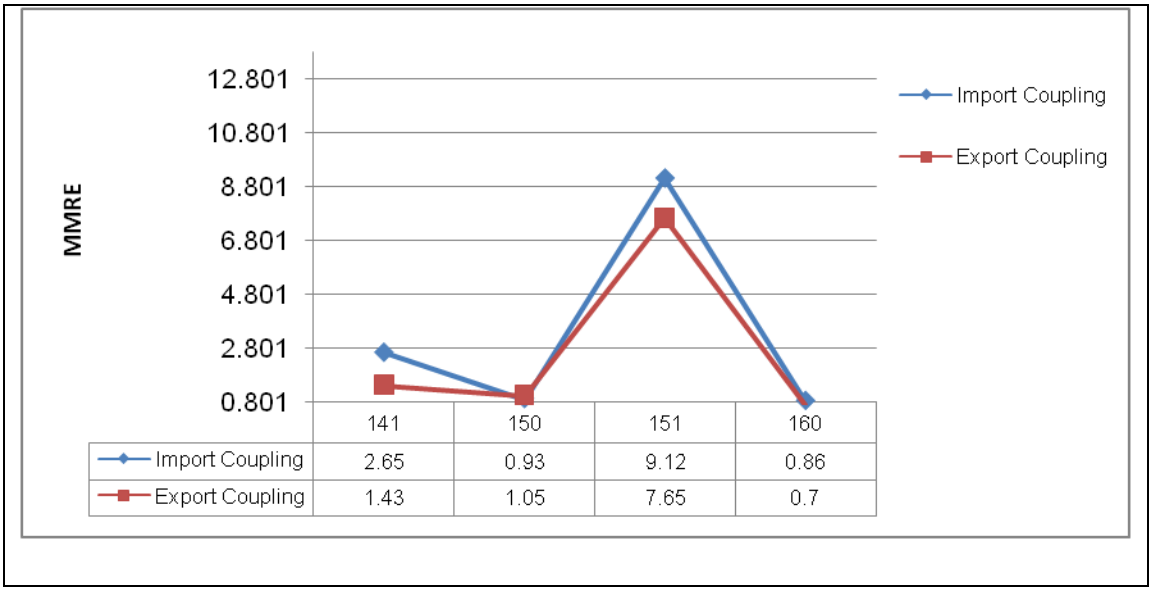


Figure 19: MMRE curves for import coupling metrics and export coupling metrics model for LabPlot

In release 071, 081 and 090 of Stellarium, the export coupling metrics model had better MMRE values than the import coupling metrics model. In release 080, the import coupling model was better, but in release 082 both had almost similar results. The average MMRE value for export coupling metrics was 6.85 while it was 7.21 for the import coupling metrics model.

In LabPlot, the MMRE values for the export coupling metrics model was better than for the import coupling metrics model in three releases. The average MMRE value for the export coupling metrics model was 2.71 while it was 3.39 for the import coupling metrics model. In both systems in general, export coupling metrics model outperforms import coupling metrics model.

As a result of the analysis, the null hypothesis of Hypothesis 3 is accepted (that is, the accuracy of import coupling metrics is lower than or equal to the accuracy of

export coupling metrics in predicting the change-density of classes in evolving object-oriented software systems).

The same analysis was performed again, but this time the pre-processing step was applied to the metrics before they entered the model. Figure 20 and Figure 21 present the result for Stellarium and LabPlot respectively.

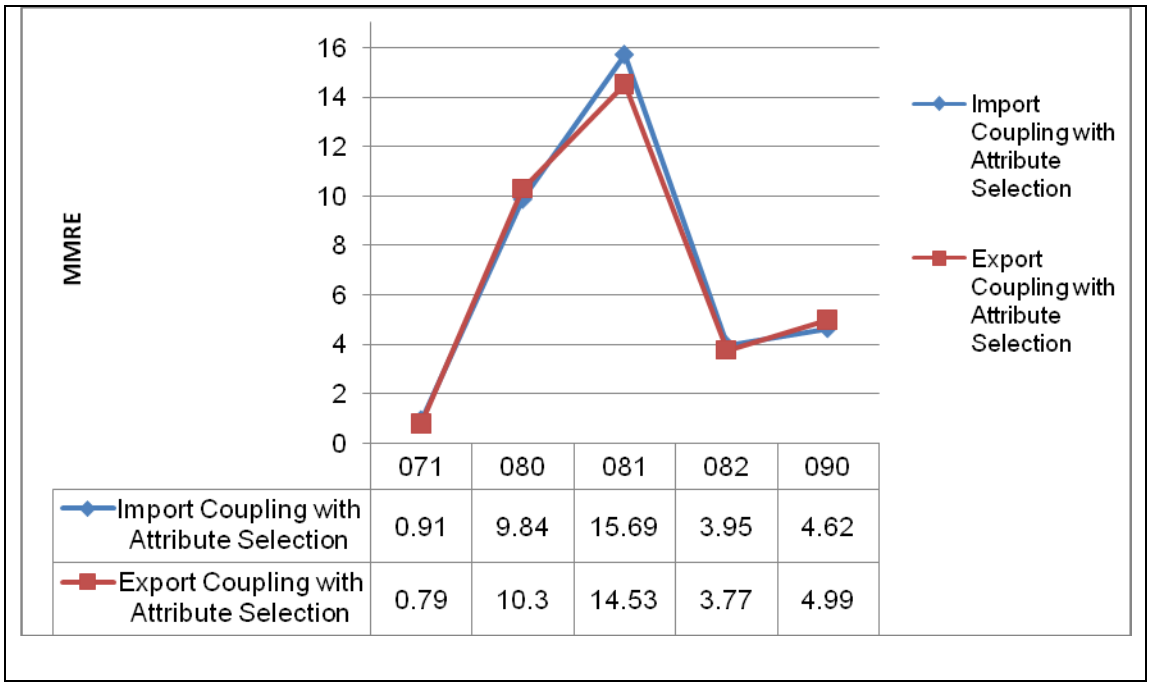


Figure 20: MMRE curves for (subset) import coupling metrics and (subset) export coupling metrics model for Stellarium

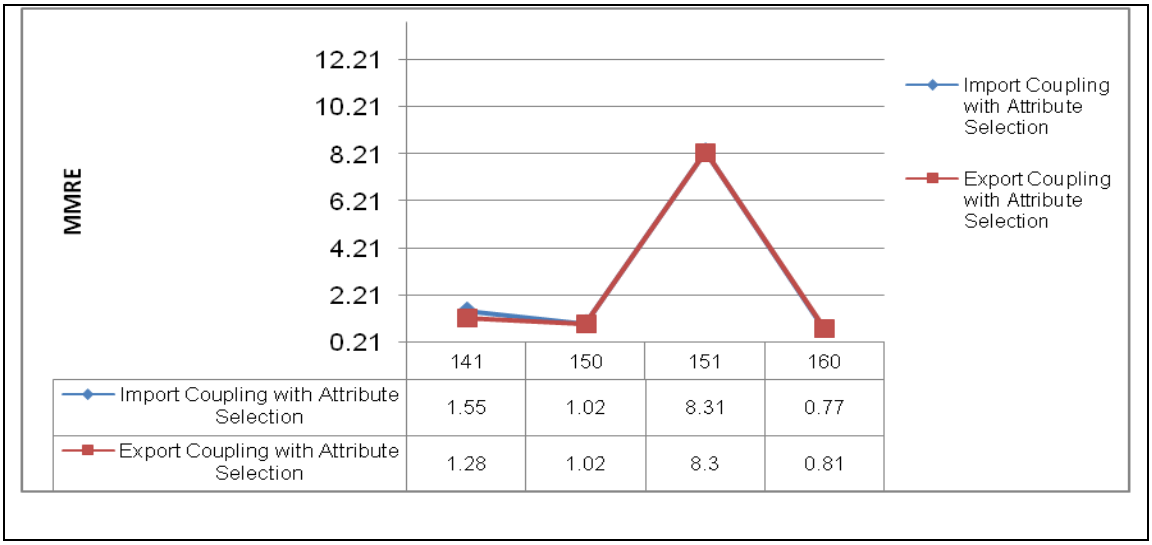


Figure 21: MMRE curves for (subset) import coupling metrics and (subset) export coupling metrics model for LabPlot

It can be observed that in both systems the import and the export coupling metrics models performed almost the same in terms of MMRE values. Also, the results of PRED(0.25) in Appendix C showed that both models behaved the same. The average MMRE values for import and export, based on pre-processing for Stellarium, were 6.88 and 7.0 respectively. The average MMRE values for import and export, based on pre-processing for LabPlot, were 2.85 and 2.91 respectively.

When comparing the models based on all metrics and models based on pre-processing (refer to Table 24 and Table 25), it can be observed that the results are almost similar. In Stellarium, the import coupling metrics model became better in terms of average MMRE and the number of points with pre-processing, while it was stable in the export coupling metrics model.

In LabPlot, the import coupling metrics model improved with pre-processing while the average MMRE value for the export coupling metrics model with pre-

processing increased. However, the difference of average MMRE between the model based on all metrics and the model based in pre-processing was to some extent small. For both systems in general, pre-processing step provided almost similar results as with all metrics.

	Average MMRE		Number of Times Model Is Better	
	All Metrics	Subset Selection	All Metrics	Subset Selection
Import Coupling Model	7.21	7.0	1	2
Export Coupling Model	6.85	6.88	3	3

Table 24: Comparison between models built on all metrics and models built on subset of metrics for Stellarium

	Average MMRE		Number of Times Model Is Better	
	All Metrics	Subset Selection	All Metrics	Subset Selection
Import Coupling Model	3.39	2.91	1	1
Export Coupling Model	2.71	2.85	3	1

Table 25: Comparison between models built on all metrics and models built on subset of metrics for LabPlot

5.2.2.3 Model Built Based on Each Metric in C&K Suite

The aim in this analysis is to compare the capability of the coupling metrics in the C&K suite in predicting the change-density of classes with other metrics in the same suite. When coupling metrics in this suite show better or similar result with other metrics, this can support the use of coupling metrics in predicting change-density of classes.

In Stellarium, six models were created for each metric in the C&K suite to predict the change-density of classes. For LabPlot, however, models for DIT and NOC were not created since inheritance was not used in LabPlot. Table 26 and Table 27 show the

MMRE results for Stellarium and LabPlot respectively. The boldface values indicate the minimum MMRE among other metrics in the specified release.

Releases	MMRE						
	071	080	081	082	090	*AVG*	*Number a model was better*
CBO	0.93	10.72	16.72	3.9	4.32	*7.32*	*0*
RFC	0.89	11.26	15.96	3.96	4.37	*7.29*	*0*
LCOM	**0.83**	11.05	14.56	4.29	4.39	*7.02*	*1*
WMC	0.89	10.25	**13.76**	3.97	**4.06**	***6.59***	*2*
NOC	0.92	9.93	14.67	3.83	4.3	*6.73*	*0*
DIT	0.98	**9.76**	14	**3.75**	4.44	***6.59***	*2*

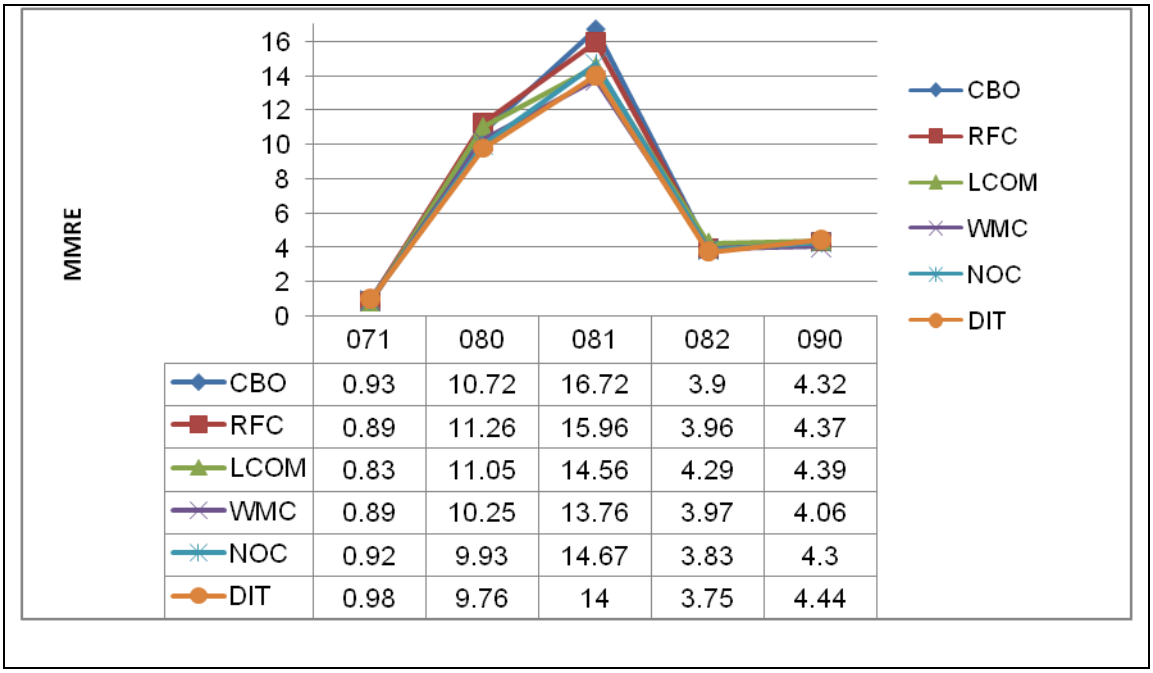
Table 26: MMRE for each metric in the C&K suite for Stellarium

Releases	MMRE					
	141	150	151	160	*AVG*	*Number a model was better*
CBO	**1.28**	1.02	8.39	0.82	*2.88*	*1*
RFC	1.31	1.03	8.42	0.82	*2.90*	*0*
LCOM	1.29	1.03	8.55	0.84	*2.93*	*0*
WMC	1.36	**0.97**	**8.21**	**0.76**	***2.83***	*3*

Table 27: MMRE for each metric in the C&K suite for LabPlot

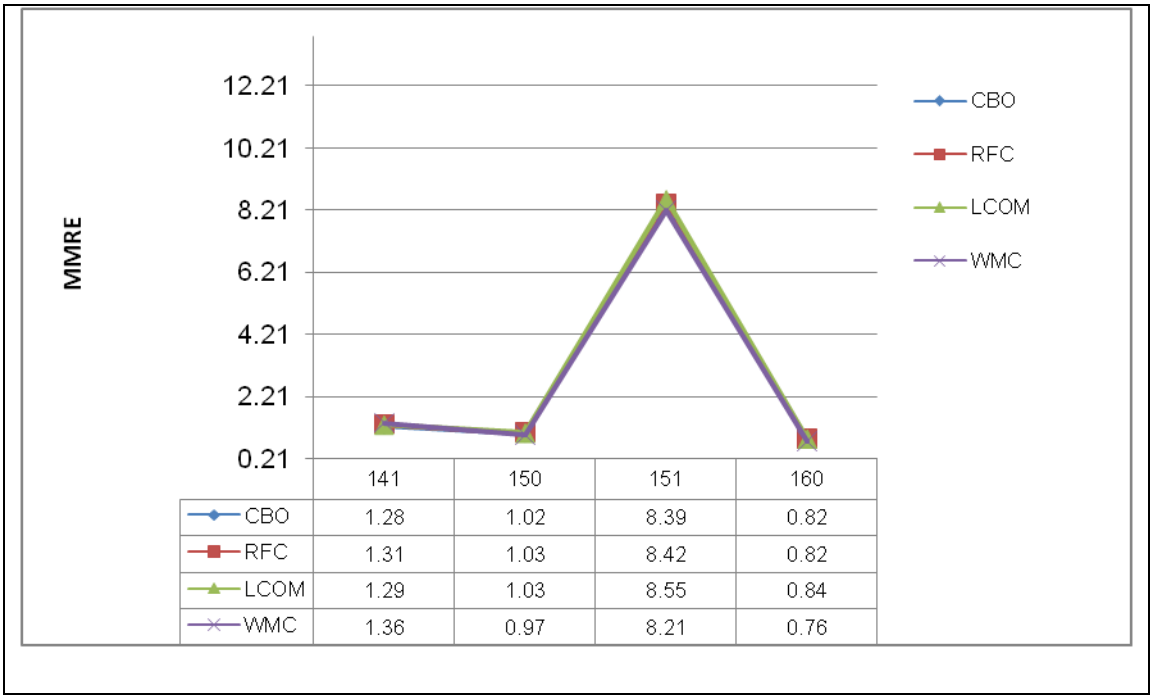
Figure 22 and Figure 23 show the MMRE curve for each metric in the C&K suite for Stellarium and LabPlot respectively. As it can be noted, all models in both systems seem to follow a similar pattern. This may indicate that the coupling metrics within the C&K suite follow similar behavior as other metrics in the suite.

As a result of the above analysis, the null hypothesis of Hypothesis 4 is accepted (that is, coupling metrics in the C&K suite provide lower or same accuracy as other metrics in the same suite in predicting the change-density of class in an evolving object-oriented software system).



	071	080	081	082	090
CBO	0.93	10.72	16.72	3.9	4.32
RFC	0.89	11.26	15.96	3.96	4.37
LCOM	0.83	11.05	14.56	4.29	4.39
WMC	0.89	10.25	13.76	3.97	4.06
NOC	0.92	9.93	14.67	3.83	4.3
DIT	0.98	9.76	14	3.75	4.44

Figure 22: MMRE curves for each model created for each metric in the C&K suite for Stellarium



	141	150	151	160
CBO	1.28	1.02	8.39	0.82
RFC	1.31	1.03	8.42	0.82
LCOM	1.29	1.03	8.55	0.84
WMC	1.36	0.97	8.21	0.76

Figure 23: MMRE curves for each model created for each metric in the C&K suite for LabPlot

5.2.3 Confounding Effect of Class Size

The analysis in section 4.2.5 was followed but this time to explore whether the association between the investigated coupling metrics and change-density of classes (refer to section 5.2.1 for more detail) was real or not.

Table 28 and Table 29 show the results for Stellarium and LabPlot respectively after controlling the size. These results reflect only those metrics found to be significant in univariate analysis (section 5.2.1). The bolded values in the tables indicate those metrics whose p-value is less than 0.05.

Coupling Metrics	Z	p-value
CBO	0.916528	0.359390
CBO1	**2.468442**	**0.013571**
RFC	**2.709699**	**0.006735**
RFC1	**2.709699**	**0.006735**
MPC	**2.697098**	**0.006995**
DAC	**3.387656**	**0.000705**
DAC1	**2.686373**	**0.007224**
ICP	**3.023263**	**0.002501**
NIH-ICP	**2.847801**	**0.004403**
OCAIC	**3.238652**	**0.001201**
OCAEC	**4.132614**	**0.000036**
OCMIC	**5.505184**	**0.000000**

Table 28: Result of the model after controlling the size for Stellarium

Coupling Metrics	Z	p-value
CBO1	1.695239	0.090031
DAC	1.556918	0.119491
DAC1	1.655637	0.097796
OCAIC	1.556918	0.119491

Table 29: Result of the model after controlling the size for LabPlot

In Stellarium, the CBO was found to have a real association with the change-density of classes, while in other metrics, after controlling for the size confounding, the association disappears. For LabPlot, all metrics which showed a significant association

with the change-density of classes in univariate analysis were found to have no confounding effect of class-size.

The association between some of the investigated coupling metrics and the change-density of classes is real, since some of these metrics are shown to be significant after controlling the class size. Therefore, the null hypothesis of Hypothesis 6 is partially rejected, and the alternative hypothesis is partially accepted (that is, there is no confounding effect of class size in the validity of the some investigated coupling metrics).

5.2.4 Threats to Validity

The threats to validity in section 4.2.6 are also applicable to empirical study II. In this empirical study, however, a distinction was made between changes caused in the systems by the density of changes, whereas in empirical study I all changes were considered equivalent.

Chapter 6

Conclusion

In this research, empirical validation of coupling metrics was performed to explore their capability to identify change-prone classes and to predict the change-density of classes in an evolving object-oriented software system. Changeability of a system's class was considered in two forms: (i) changed or not changed from one release to the next; and (ii) the density of changes in a class from one release to the next. Analyses were performed to investigate the relationship between an individual coupling metric, when all used together and when only subset are used with changeability. Several models were created on the basis of regression techniques. Several findings were obtained from these analyses:

- The investigated coupling metrics models were found to be good predictors of the change-proneness and change-density of classes. The accuracy of the investigated coupling metrics models was better or comparable with the cohesion metrics model.
- Applying subset selection to the coupling metrics before creating the prediction model is helpful, and it can provide similar results as with all coupling metrics.

- Coupling metrics in the C&K suite were found to provide comparable results for predicting change-proneness and change-density.
- Import coupling metrics models were found to be better predictors of change-proneness of classes than the export coupling metrics models.
- There is no confounding effect of class size in the validity of some of the investigated coupling metrics. In other words, there is a real association between some of the investigated coupling metrics and the change-proneness and change-density of classes.

6.1 Major Contribution

The main contributions of the thesis are summarized as follows:

- Empirical validation of a comprehensive set of coupling metrics was performed to explore the capability of coupling metrics to (i) identify change-prone classes; and (ii) predict the change-density of classes in a context of an evolving object-oriented software system.

- Several models were built to identify the change-prone classes and to predict change-density of classes, and the accuracies of these models were compared. These models are:

    - Model based on coupling metrics and model based on subset of coupling metrics,

    - Model based on cohesion metrics and model based on subset of cohesion metrics,

- Models based on each metric in C&K suite,
- Model based on import coupling metrics and model based on subset of import coupling metrics, and
- Model based export coupling metrics and model based on subset of export coupling metrics.

- The confounding effect of class size on identifying the change-prone classes and predicting the change-density of classes was investigated.

6.2 Future Work

Coupling metrics were found to be good indicators of the changeability of an evolving software system. Exploring the relationship between coupling metrics and other software quality attributes such as testability, reusability, and etc is a good area for future research.

Coupling metrics investigated in this study were static coupling metrics. Investigating the capability of dynamic coupling metrics, and comparing them with static coupling metrics, are important areas of research.

Another important study which can be made in future is to study the capability of coupling metrics to predict changeability with other object-oriented structural properties such as inheritance or cohesion. It might be helpful to investigate models based on more than one structural property.

Finally, the analysis performed in this research was based on simple regression methods. More sophisticated machine learning techniques such as Artificial Neural Networks (ANNs) and decision trees can be employed to determine their effectiveness in predicting the changeability.

Bibliography

[1] A. Günş Koru, Liu Hongfang, "Identifying and characterizing change-prone classes in two large-scale open-source products," Journal of Systems and Software, vol. 80, pp. 63-73, 2007

[2] D. Poshyvanyk and A. Marcus, "The Conceptual Coupling Metrics for Object-Oriented Systems," *22nd IEEE Int. Conf. on Software Maintenance (ICSM'06)*, pp. 469-478, 2006

[3] E. Arisholm, , Briand, L. C., and Foyen, A., "Dynamic coupling measurement for object-oriented software", *IEEE Transactions on Software Engineering*, vol. 30, no. 8, pp. 491-506, August 2004

[4] E. Arisholm, "Empirical Assessment of the Impact of Structural Properties on the Changeability of Object-Oriented Software," *Information and Software Technology (48),* pp. 1046-1055, 2006

[5] FrontEndART Software Ltd., http://www.frontendart.com, 2005. User Guide For Columbus

[6] F. Abreu, M. Goulão, and R. Esteves, "Toward the Design Quality Evaluation of Object-Oriented Software Systems," *Proc. Fifth Int'l Conf. Software Quality*, Austin, Texas, pp. 44-57, October 1995.

[7] G. Gui and P. D Scott, "Ranking Reusability of software Components Using Coupling Metrics," *Journal of Systems and Software*, pp. 1450-1459, September 2006

[8] G. Dunteman, Principal Component Analysis. Sage University Paper 07-69, Thousand Oaks, CA. 1989

[9] F. G. Wilkie and B.A. Kitchenham, "An Investigation of Coupling, reuse and Maintenance in a Commercial C++ Application," *Information and Software Technology*, vol. 43, pp. 801-812, 2001

[10] H. M. Olague, L.H. Etzkorn, S. Gholston, and S. Quattlebaum, " Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes," *IEEE Transactions on Software Engineering*, vol. 33, issue 6, pp. 402-419, June 2007

[11] I. Deligiannis, M. Shepperd, M. Roumeliotis and I. Stamelos, “An Empirical Investigation of an Object-Oriented Design Heuristic for Maintainability,” *Journal of Systems and Software*, vol. 65, pp. 127-139, 2003

[12] I. H. Witten and E. Frank, *Data Mining Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publisher, 2005

[13] I. Sommerville , *Software Engineering*, Addison-Wesley, 2001

[14] J. Eder, G. Kappel, and M. Schrefl, “Coupling and Cohesion in Object-Oriented Systems,” Technical Report, Univ. of Klagenfurt, 1994

[15] K. El Emam, W. Melo, and J. Machado, “The Prediction of Faulty Classes Using Object-Oriented Design Metrics,” *Journal of Systems and Software*, vol. 56, pp. 63-75, 2001

[16] K. El Emam, S. Benlarbi, N. Goel, and S.N. Rai, “The Confounding Effect of Class Size on the Validity of Object-Oriented Metrics,” *IEEE Transactions on Software Engineering*, vol. 27, pp. 630-650, 2001

[17] L. Briand, P. Devanbu, and W. Melo, “An Investigation into Coupling Measures for C++,” *Proc. 19th Int’1 Conf. Software Eng.*, ICSE’97, Boston, pp.412-421, May 1997

[18] L. Briand, J. Daly, V. Porter, and J. Wüst, “Predicting Fault-prone Classes with Design Measures in Object-Oriented Systems,” *Proc. Nith Int’1 Symp. Software Reliability Eng. ISSRE’98*, Paderborn Germany, pp. 334-343, November 1998

[19] L. C. Briand, J. W. Daly, and J. K. Wüst, "A Unified Framework for cohesion measurement in object-oriented systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 65-117, 1998

[20] L. C. Briand, J. W. Daly, and J. K. Wüst, “A Unified Framework for Coupling Measurement in Object-Oriented Systems,” *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91-121, 1999

[21] L. Briand, J. Daly, V. Porter, and J. Wüst, “Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems,” *Journal of Systems and Software*, vol. 51, pp. 245-273, 2000

[22] L. Zhao, S. Elbaum, "Quality assurance under the open source development model," *Journal of Systems and Software*, vol. 66, no. 1, pp. 65-75, 2003

[23] M. Bruntink and A. Deursen, "An empirical study into class testability," *Journal of Systems and Software*, vol. 79, pp. 1219-1232, 2006

[24] M. Hitz and B. Montazeri, “Measuring Product Attributes of Object-Oriented Systems,” W. Schöfer and P. Botella, eds., *Proc. ESEC ‘95 Fifth European Software Eng. Conf.*, Barcelona, Spain, pp. 124-136, September 1995

[25] N.I. Churcher and M.J. Shepperd, “Towards a Conceptual Framework for Object Oriented Software Metrics,” *Software Engineering Notes*, vol. 20, no. 2, pp. 69-76, 1995

[26] P. Joshi and R. K. Joshi, “Microscopic Coupling Metrics for Refactoring,” *Pro. Of the Con. On Software Maintenance and Reengineering (CSMR’06)*, pp. 145-152, 2006

[27] R. C. Seacord, D. Plakosh, G. A. Lewis, *Modernizing Legacy Systems* , Addison-Wesley, 2003

[28] R. Harrison, S. Counsell and R. Nithi, “Coupling Metrics for Object-Oriented Design,” *Proceedings. Fifth International Symposium on Software Metrics*, vol. 20. , issue 21, pp. 150 – 157, November 1998

[29] R. S. Pressman, *Software Engineering A Practitioner's Approach*, McGraw-Hill, 2005

[30] R. Subramanyam and M.S. Krishnan, "Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects," *IEEE Transactions on Software Engineering*, vol. 29, no. 4,pp. 297-310, April 2003

[31] S. R. Chidamber and C.F. Kemerer, “Towards a Metrics Suite for Object Oriented Design,” A. Paepcke, ed., *Proc. Conf. Object-Oriented Programming: Systems, Languages and Applications, OOPSLA’91*, Oct. 1991. Also published in *SIGPLAN Notices*, vol. 26, no. 11, pp. 197-211, 1991

[32] S. R. Chidamber and C.F. Kemerer, “A Metrics Suite for Object Oriented Design,” *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, 1994

[33] T. Gyimóthy, R. Ferenc, and I. Siket, “Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Predication,” *IEEE Transactions on Software Engineering*, vol. 31, no. 10, October 2005

[34] W. Li and S. Henry, “Object-Oriented Metrics that Predict Maintainability,” *Journal of Systems and Software*, vol. 23, no. 2, pp. 111-122, 1993

[35] W. Stevens, G. Myers, and L. Constantine, “Structured Design,” *IBM Systems Journal*, vol. 13, no. 2, pp. 115-139, 1974

[36] Y. Liu and A. Milanova, “Static Analysis for Dynamic Coupling Measures,” Pensselaer Polytechnic Institute, Troy, NY 12180, USA, 2006

[37] Y.-S. Lee, B.-S. Liang, S.-F. Wu, and F.-J. Wang, “Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow,” *Proc. Int’l Conf. Software Quality*, Maribor, Slovenia, pp. 81-90, 1995

Appendix A: Metrics Used in the Research

Table 31 and Table 32 describe the cohesion metrics and the C&K metrics suite used in this study. In each table, the column "Name" states the acronym of each metric and what it stands for. The "Definition" column provides an informal natural definition of the metric. These should give the reader a quick insight into the measure. Formal definitions of the measures, with a uniform and unambiguous formalism, are provided in [19][20][31][32].

Name	Definition	Source
LCOM1 (lack of cohesion in methods)	The number of pairs of methods in the class using no attribute in common.	[31][32]
LCOM2	The number of pairs of methods in the class using no attributes in common, minus the number of pairs of methods that do. If this difference is negative LCOM2 is set to zero.	[31][32]
LCOM3	Consider an undirected graph *G*, where the vertices are the methods of a class, and there is an edge between two vertices if the corresponding methods use at least an attribute in common. LCOM3 is defined as the number of connected components of *G*.	[19]
LCOM4	Like LCOM3, where graph *G* additionally has an edge between vertices representing methods *m* and *n*, if *m* invokes *n* or vice versa.	[19]
LCOM5	Consider a set of methods {*Mi*} (*i*=1,...,*m*) accessing a set of attributes {*Aj*} (*j*=1,...,*a*). Let μ(*Aj*) be the number of methods which reference attribute *Aj*. Then $LCOM5 = \left(\frac{1}{a} \left(\sum_{j=1}^{a} \mu(A_j) \right) - m \right) / (1 - m)$	[19]

ICH (information flow-based cohesion)	ICH for a method is defined as the number of invocations of other methods of the same class, weighted by the number of parameters of the invoked method (cf. coupling measure ICP). The ICH of a class is the sum of the ICH values of its methods.	[19]
TCC (tight class cohesion)	Besides methods using attributes directly (by referencing them), this measure considers attributes indirectly used by a method. Method *m* uses attribute *a* indirectly, if *m* directly or indirectly invokes a method which directly uses attribute *a*. Two methods are called connected, if they directly or indirectly use common attributes. TCC is defined as the percentage of pairs of public methods of the class which are connected, i.e. pairs of methods which directly or indirectly use common attributes.	[19]
LCC (loose class cohesion)	Same as TCC, except that this measure also considers pairs of indirectly connected methods. If there are methods *m1*,...,*mn*, such that *mi* and *mi*+1 are connected for *i*=1,...,*n*-1, then *m1* and *mn* are indirectly connected. Measure LCC is the percentage of pairs of public methods of the class which are directly or indirectly connected.	[19]
Coh (Cohesion)	A variation on LCOM5 $Coh = \left(\sum_{j=1}^{a} \mu(A_j)\right)/(m \cdot a)$	[19]
Co (connectivity)	Let *V* be the number of vertices of graph *G* from measure LCOM4, and *E* the number of its edges. Then $Co = 2(\lvert E\rvert - (\lvert V\rvert - 1))/((\lvert V\rvert - 1)(\lvert V\rvert - 2))$	[19]

Table 30: Cohesion metrics

WMC (weighted method per class)	Weighted methods for class. The weight is the McCabe cyclomatic complexity.	[21][31][32]
LCOM1(lack of cohesion in method)	The number of pairs of methods in the class using no attribute in common.	[21][31][32]
CBO (coupling between object)	A class is coupled to another, if methods of one class use methods or attributes of the other, or vice versa.	[21][31][32]
RFC (Response for Class)	The response set of a class consists of the set *M* of methods of the class, and the set of methods directly or indirectly invoked by methods in *M*.	[21][31][32]
DIT (depth of inheritance tree)	The DIT of a class is the length of the longest path from the class to the root in the inheritance hierarchy.	[21][31][32]
NOC (number of children)	The number of classes that directly inherit from a given class.	[21][31][32]

Table 31: C&K suite

Appendix B: Metrics Resulting from Pre-processing Step

Table 32 to Table 39 show the resulting metrics after applying the pre-processing step for both systems for change-proneness. Table 40 to Table 47 show the resulting metrics after applying the pre-processing step for both systems for change-density.

System releases	Resulting metrics
0.7.1	CBO1 MPC DAC NIH-ICP OCMIC
0.8.0	CBO1 MPC DAC NIH-ICP OCMIC
0.8.1	CBO1 RFC1 MPC NIH-ICP OCAIC OCMIC
0.8.2	CBO1 RFC RFC1 IH-ICP NIH-ICP OCAIC OCMIC OCMEC
0.9.0	CBO1 RFC RFC1 DAC IH-ICP NIH-ICP OCAIC OCMIC OCMEC

Table 32: Resulting coupling metrics for Stellarium for change-proneness

System releases	Resulting metrics
1.4.1	DAC OCMIC
1.5.0	RFC DAC DAC1 OCMIC OCMEC
1.5.1	RFC RFC1 DAC DAC1 OCAIC OCAEC OCMIC OCMEC
1.6.0	RFC DAC OCAEC OCMIC

Table 33: Resulting coupling metrics for LabPlot for change-proneness

System releases	Resulting metrics
0.7.1	LCOM1 LCOM4 LCOM5 ICH Coh
0.8.0	LCOM1 LCOM2 LCOM5 ICH Coh
0.8.1	LCOM1 LCOM2 LCOM5 Coh
0.8.2	LCOM1 LCOM2 LCOM5 Coh
0.9.0	LCOM2 LCOM5 Coh

Table 34: Resulting cohesion metrics for Stellarium for change-proneness

System releases	Resulting metrics
1.4.1	LCOM1
1.5.0	LCOM2 LCOM4
1.5.1	LCOM2 Coh
1.6.0	LCOM1 LCOM5 Coh

Table 35: Resulting cohesion metrics for LabPlot for change-proneness

System releases	Resulting metrics
0.7.1	RFC1 DAC IH-ICP NIH-ICP OCMIC
0.8.0	CBO1 MPC DAC NIH-ICP OCMIC
0.8.1	CBO1 RFC1 MPC NIH-ICP OCAIC OCMIC
0.8.2	CBO1 RFC RFC1 IH-ICP NIH-ICP OCAIC OCMIC
0.9.0	CBO1 RFC DAC IH-ICP NIH-ICP OCAIC OCMIC

Table 36: Resulting import coupling metrics for Stellarium for change-proneness

System releases	Resulting metrics
1.4.1	DAC OCMIC
1.5.0	RFC DAC DAC1 OCMIC
1.5.1	RFC RFC1 DAC DAC1 OCMIC
1.6.0	RFC DAC OCMIC

Table 37: Resulting import coupling metrics for LabPlot for change-proneness

System releases	Resulting metrics
0.7.1	CBO
0.8.0	CBO1
0.8.1	CBO1
0.8.2	CBO1 OCMEC
0.9.0	CBO1 OCMEC

Table 38: Resulting export coupling metrics for Stellarium for change-proneness

System releases	Resulting metrics
1.4.1	CBO
1.5.0	OCMEC
1.5.1	CBO OCAEC OCMEC
1.6.0	CBO OCAEC OCMEC

Table 39: Resulting export coupling metrics for LabPlot for change-proneness

System releases	Resulting metrics
0.7.1	RFC IH-ICP
0.8.0	CBO1 OCAEC OCMEC
0.8.1	DAC1 OCAEC OCMEC
0.8.2	CBO1 OCAEC OCMEC
0.9.0	DAC1 OCMEC

Table 40: Resulting coupling metrics for Stellarium for change-density

System releases	Resulting metrics
1.4.1	DAC1
1.5.0	CBO
1.5.1	CBO CBO1 DAC1 OCMEC
1.6.0	CBO CBO1 DAC1 OCMEC

Table 41: Resulting coupling metrics for LabPlot for change-density

System releases	Resulting metrics
0.7.1	LCOM1 LCOM3 LCOM4 LCOM5 Coh Co
0.8.0	LCOM2 TCC
0.8.1	TCC
0.8.2	LCOM5 TCC LCC
0.9.0	LCOM5 TCC

Table 42: Resulting cohesion metrics for Stellarium for change-density

System releases	Resulting metrics
1.4.1	LCOM5 Coh Co
1.5.0	LCC Co
1.5.1	LCOM5 TCC Coh
1.6.0	LCOM5 TCC Coh

Table 43: Resulting cohesion metrics for LabPlot for change-density

System releases	Resulting metrics
0.7.1	RFC IH-ICP
0.8.0	RFC1 IH-ICP
0.8.1	RFC1
0.8.2	DAC1 IH-ICP
0.9.0	DAC1 IH-ICP

Table 44: Resulting import coupling metrics for Stellarium for change-density

System releases	Resulting metrics
1.4.1	DAC1
1.5.0	CBO
1.5.1	CBO CBO1 DAC1
1.6.0	CBO1 DAC1

Table 45: Resulting import coupling metrics for LabPlot for change-density

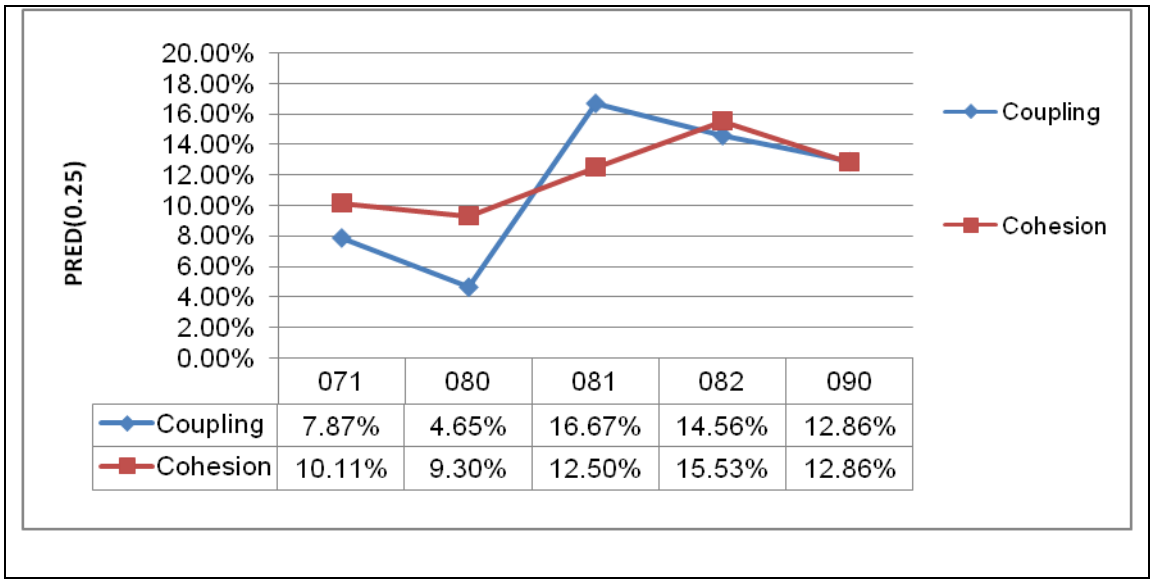
System releases	Resulting metrics
0.7.1	CBO1 OCMEC
0.8.0	CBO1 OCAEC OCMEC
0.8.1	CBO1 OCAEC OCMEC
0.8.2	CBO1 OCAEC OCMEC
0.9.0	CBO1 OCMEC

Table 46: Resulting export coupling metrics for Stellarium for change-density

System releases	Resulting metrics
1.4.1	CBO OCAEC
1.5.0	CBO
1.5.1	CBO1
1.6.0	CBO CBO1 OCMEC

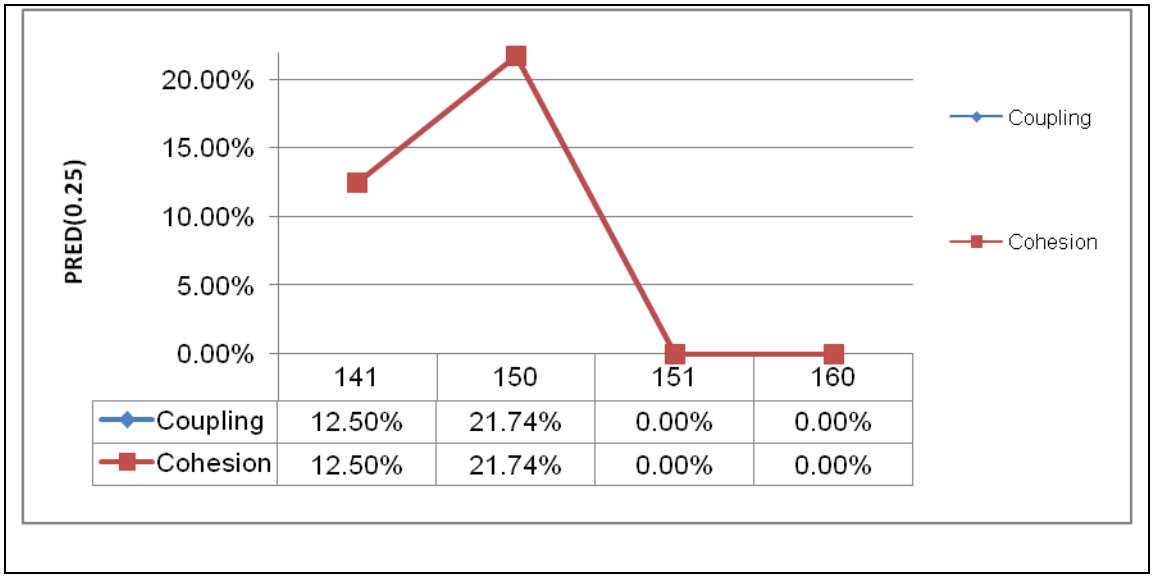
Table 47: Resulting export coupling metrics for LabPlot for change-density

Appendix C: PRED(0.25) Results for Empirical Study II



	071	080	081	082	090
Coupling	7.87%	4.65%	16.67%	14.56%	12.86%
Cohesion	10.11%	9.30%	12.50%	15.53%	12.86%

Figure 24: PRED(0.25) curves for coupling metrics and cohesion metrics model for Stellarium



	141	150	151	160
Coupling	12.50%	21.74%	0.00%	0.00%
Cohesion	12.50%	21.74%	0.00%	0.00%

Figure 25: PRED(0.25) curves for coupling metrics and cohesion metrics model for LabPlot

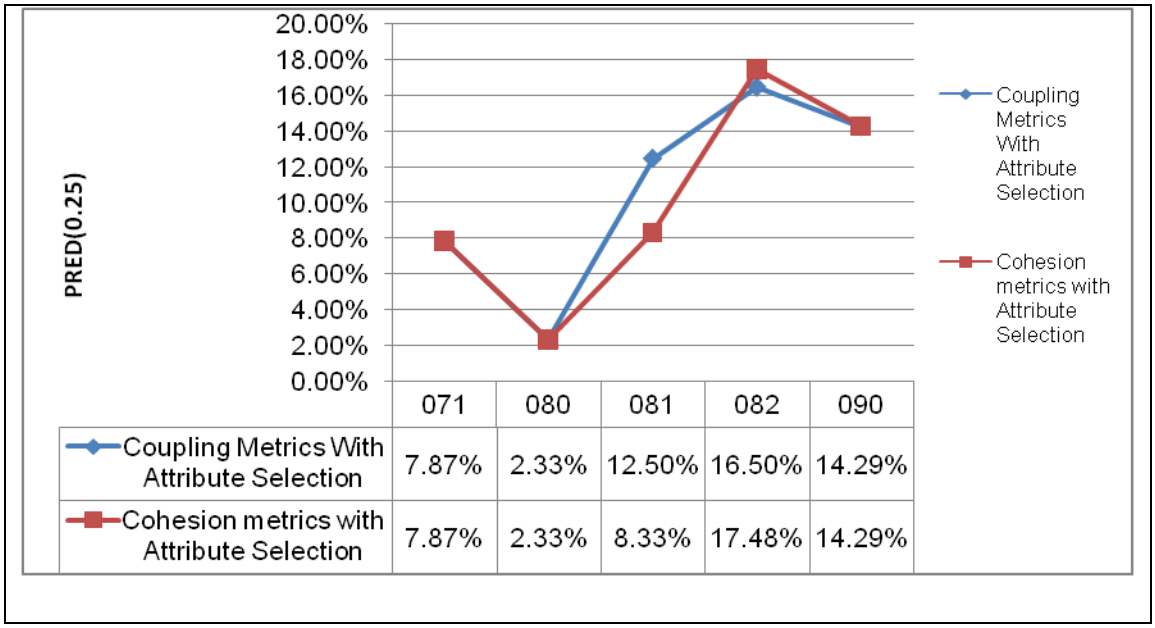


Figure 26: PRED(0.25) curves for (subset) coupling metrics and (subset) cohesion metrics model for Stellarium

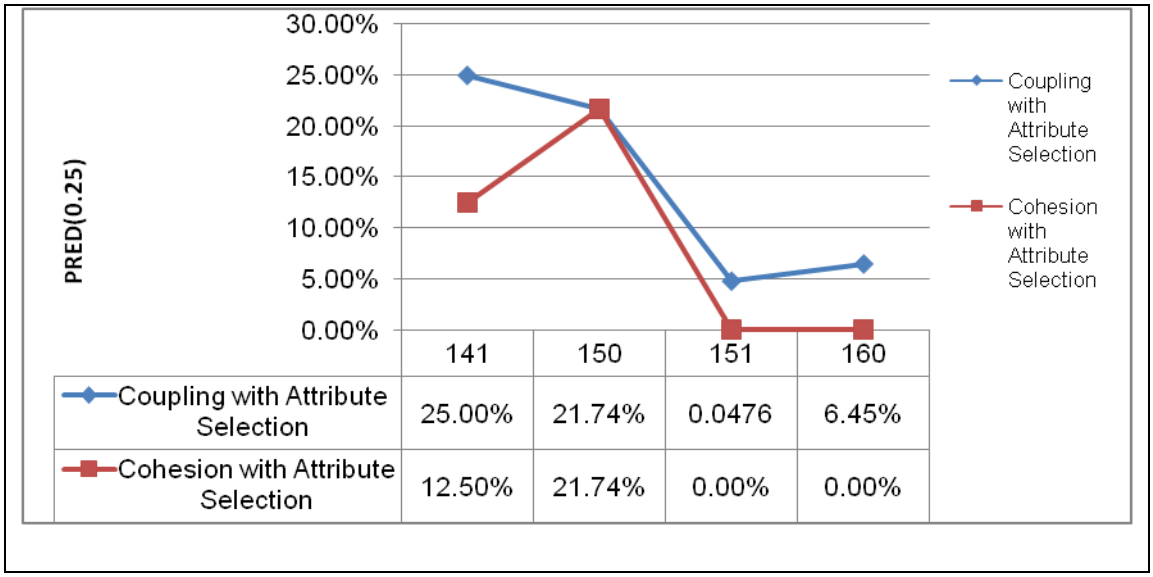


Figure 27: PRED(0.25) curves for (subset) coupling metrics and (subset) cohesion metrics model for LabPlot

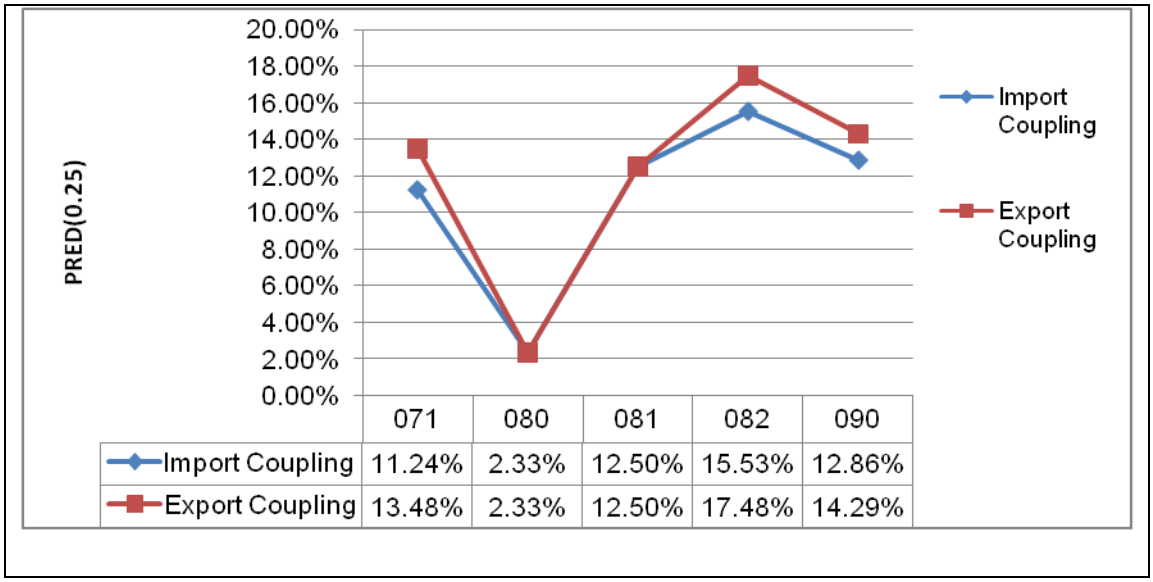


Figure 28: PRED(0.25) curves for import coupling metrics and export coupling metrics model for Stellarium

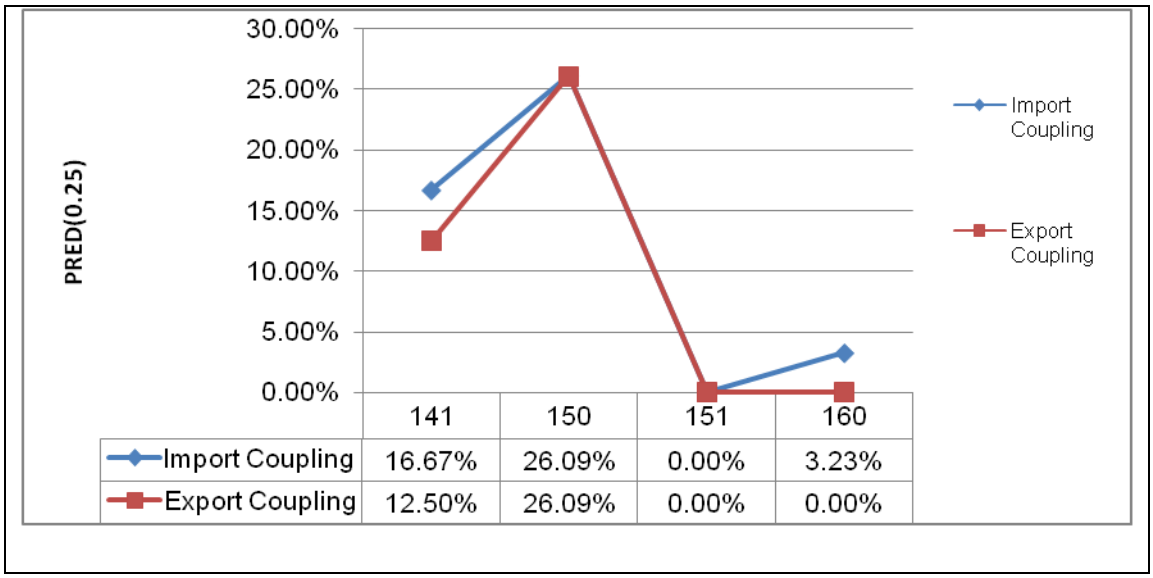
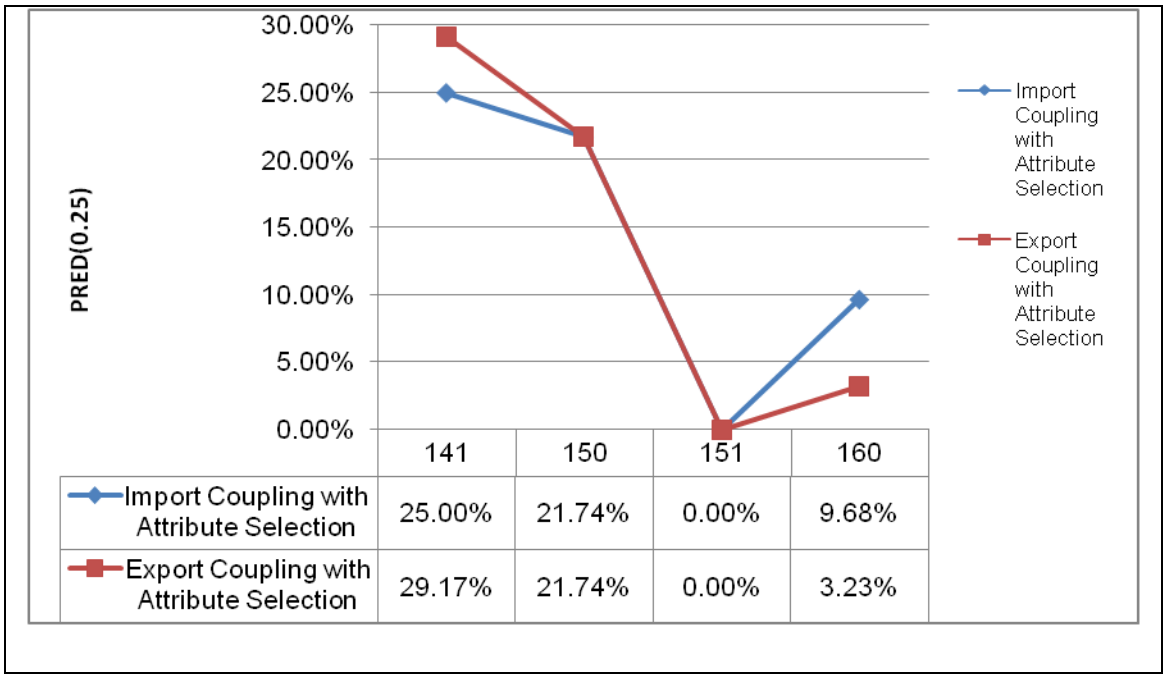


Figure 29: PRED(0.25) curves for import coupling metrics and export coupling metrics model for LabPlot

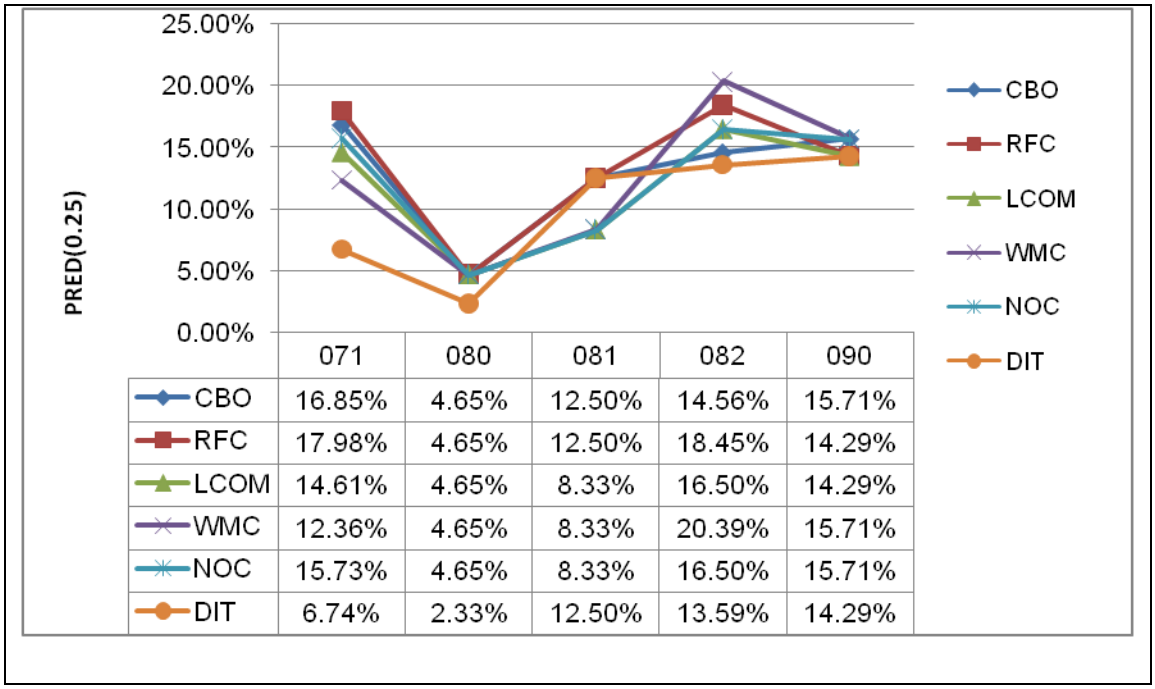
	071	080	081	082	090
Import Coupling with Attribute Selection	7.87%	4.65%	8.33%	15.53%	14.29%
Export Coupling with Attribute Selection	11.24%	2.33%	12.50%	16.50%	14.29%

Figure 30: PRED(0.25) curves for (subset) import coupling metrics and (subset) export coupling metrics model for Stellarium



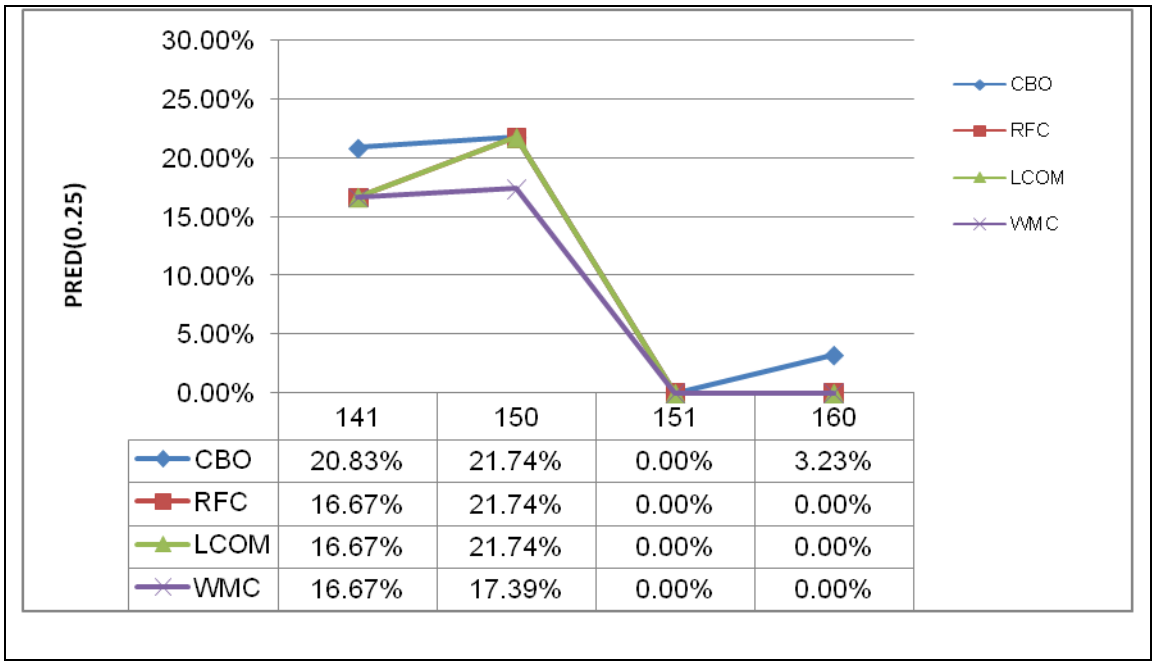
	141	150	151	160
Import Coupling with Attribute Selection	25.00%	21.74%	0.00%	9.68%
Export Coupling with Attribute Selection	29.17%	21.74%	0.00%	3.23%

Figure 31: PRED(0.25) curves for (subset) import coupling metrics and (subset) export coupling metrics model for LabPlot



	071	080	081	082	090
CBO	16.85%	4.65%	12.50%	14.56%	15.71%
RFC	17.98%	4.65%	12.50%	18.45%	14.29%
LCOM	14.61%	4.65%	8.33%	16.50%	14.29%
WMC	12.36%	4.65%	8.33%	20.39%	15.71%
NOC	15.73%	4.65%	8.33%	16.50%	15.71%
DIT	6.74%	2.33%	12.50%	13.59%	14.29%

Figure 32: PRED(0.25) curves for each metric model in the C&K suite Stellarium



	141	150	151	160
CBO	20.83%	21.74%	0.00%	3.23%
RFC	16.67%	21.74%	0.00%	0.00%
LCOM	16.67%	21.74%	0.00%	0.00%
WMC	16.67%	17.39%	0.00%	0.00%

Figure 33: PRED(0.25) curves for each metric model in the C&K suite for LabPlot