Predicting Error-Proneness of OO Projects with MOOD Metrics

Shen Li, Jiayao Zhou, Wenyu Gu, Ruijia Yang

***Abstract*—Up to date, many metric suits have been developed in order to assess software systems. Usually, they are used to measure software quality, such as maintainability, reliability, error-proneness and etc. The most notable metric suites are the Chidamber and Kemerer (CK) metrics, Abreu's Metrics for Object-Oriented Design (MOOD), and Bansiya and Davis' Quality Metrics for Object-Oriented Design (QMOOD). These metric suites have been wildly used and many people also try to validate their viability. In our study, we would like to validate if the MOOD metrics can be useful to predict error-proneness of OO projects by finding correlations between the MOOD metrics and CK metrics that have been proved to be useful as error-proneness indicator. We perform the Pearson coefficient and the linear regression to help the validation. The result shows that only some of the MOOD metrics can be useful to predict error-proneness.**

Keywords—Error-proneness, MOOD, CK, Metrics

I. Introduction

Up to date, many metric suits have been developed in order to assess software systems. Usually, they are used to measure software quality, such as maintainability, reliability, error-proneness and etc. The most notable metric suites are the Chidamber and Kemerer (CK) metrics [3], Abreu's Metrics for Object-Oriented Design (MOOD) [2], and Bansiya and Davis' Quality Metrics for Object-Oriented Design (QMOOD) [7] [8]. These metric suites have been wildly used and many people also try to validate their viability [2] [4] [5] [6].

There are different validations on the MOOD metrics and different opinions about their effectiveness. In [2] and [3], the MOOD metrics are proved to be useful to assess software quality like reliability and maintainability. However, in [5] and [6] the MOOD metrics are validated to be not effectively for predicting software quality and the majority of the MOOD metrics being fundamentally flawed. In our study, we would like to validate if the MOOD metrics can be used to predict software quality. Since the CK metrics proposed in [3] are proved to be useful as quality indicators in terms of error-proneness, we are going to examine that if any correlations exist between these metrics and the MOOD metrics. If so, we can say that the MOOD metrics can be useful for predict error-proneness.

The following sections in this paper is organized as follows. Section 2 introduces the work that are related to our topic and the comparison with our study. Section 3 presents the metrics we consider in our study and their formal definitions. Section 4 shows the empirical study we conduct, describes the variables, the hypotheses, the experiments and the statistical tests involved in our study and discusses the findings depending on the results. Section 5 concludes the output of our study.

II. Related Work

The need for object-oriented metrics is increasing since managers are increasingly focusing on process improvement in the software development area. In [3], the author addressed this need through developing and implementing a suite of metrics for Object-Oriented (OO) design known as Chidamber and Kemerer (CK) metrics. These metrics were found to process a number of desirable properties to reflect essential OO design features. In addition, [3] presented empirical study on these metrics from actual commercial systems that are implemented by C++ and Smalltalk. The results demonstrated the metric suite provides senior designers and managers with an indication of the integrity of the design. They can use it to address the architectural and structural consistency of the entire application and identify areas of the application that may require more rigorous testing and areas that are candidates for redesign. In our study, we are going to use the CK metrics proposed in [3]. Unlike [3] focusing on developing and proving the viability of the CK metrics in OO design measurements, we use the CK metrics to measure error-proneness of software. The capability of the CK metrics in this measurement has been proved effectively in [1]. In [3], the author concludes that the CK metrics may be useful in OO design measurement. Since OO design measurement relates to many different areas such as maintainability, reusability and reliability, the conclusion of [3] is too broad. In our study, one of our focus is error-proneness. We will find correlations between values of the CK metrics and the MOOD metrics to test if the MOOD metrics can also be used to indicate error-proneness.

In [1], the research was based on the background that the requirements for metrics which reflect the specificities of the OO paradigm increased. The work in [1] was an additional step an experimental validation of the OO metric suite defined in [3]. In [1], the author presented the results of a study in which they performed an experimental validation of that suite of OO metrics with regard to their ability to identify fault-prone classes. They collected data during eight medium-sized management information system’s development. They used the same lift cycle model, analysis/design method and programming language. Through analyzing the results, they concluded that five metrics: Weighted Methods Per Class(WMC), Depth of Inheritance (DIT), Response For a Class (RFC), Number of Children (NOC) and Coupling Between Objects (CBO) are proved to be very useful as quality indicators in terms of error-proneness. Furthermore, these metrics appear to be complementary indicators which are relatively independent from each other. Depending on the conclusion of [1], we can know that WMC, DIT, RFC, NOC and CBO are proved to be very useful as quality indicators in terms of error-proneness. In our study, we will use four of these metrics to measure error-proneness of software. We are going to compute these metrics and the MOOD metrics on the same software and try to find correlations between the results to prove the MOOD metrics can also be used as indicator of error-proneness. In [1], the systems used for experiments are small and their complexity is limited. In our study, we use JFreeChart, which is a big open source project that has more than 50 versions, for our experiments.

Due to the importance of theoretical validation of object-oriented software metrics, [2] described an experimental process collecting data to support the theoretical validation of a set of metrics for object-oriented design called the MOOD metrics. The authors used three different projects to provide data. They considered a number of criteria for valid metrics to give the investigation strengths. Then [2] presented an empirical validation of the MOOD metrics under the headings encapsulation, inheritance, coupling and polymorphism. The authors used the MOOD metrics to measure three applications from different domains. The results showed that the MOOD metrics provided useful overall assessment which can be used to analysis important information for managers of software development projects to make decisions. From [2], we learn that the MOOD metrics can be used to assess properties of software systems. In our study, we will implement the MOOD metrics depending on the definitions in [2] and test if they can be used as quality indicators. In [2], it proves that the MOOD metrics are useful to assess software. Similar to [3], this conclusion is too broad. In our study, we are going to validate if the MOOD metrics are capable of indicating software quality in terms of error-proneness and change-proneness.

The main goal of [4] is to evaluate the impact of OO design on software quality characteristics. In order to measure the OO design characteristics, the MOOD metrics was adopted. In [4], the author collected both product and process data in a controlled experiment on the development of eight small-sized information management systems based on identical requirements were used to assess the referred impact and evaluate the impact of OO design on software quality by examining the degree to which the MOOD metrics allow to predict defect density and normalized rework. Some linear regression models that allow to predict the cumulative impact of all MOOD metrics on resulting software quality characteristics are applied and validated. In [4], it shows that Attribute Hiding Factor (AHF) and Coupling Factor (COF) are bigger contributors to predictive model, while Method Inheritance Factor (MIF) and Attribute Inheritance Factor (AIF) have lower influence. However, even the latter metrics have a sufficient impact on the model not to be considered as extraneous. And the models could produce estimates where only around 1.27% of the defect density, 23.24% of the failure density and 0.03% of the normalized rework effort will be left unexplained. The results show that the design alternatives may have a strong influence on resulting quality. Quantifying this influence can help to train novice designers by means of heuristics embedded in design tools. Being able to predict the resulting reliability and maintainability is very important to project managers during the resource allocation (planning) process. In [4], the author adopts the MOOD metrics to measure the characteristics of OO design. We also use these metrics in our study. But we are trying to examine if the MOOD metrics are useful to indicate error-proneness and change-proneness of software instead of focusing on reliability and maintainability like [4]. The limitation of [4] is that the authors assume the MOOD metrics are useful to measure reliability and maintainability which is not proved. In our study, we use the CK metrics to measure error-proneness since [1] proves that the CK metrics are capable of this measurement.

Some CK class metrics have previously been shown to be good predictors of initial OO software quality. However, the other two metric suites MOOD and QMOOD have not been heavily validated except by their original proposers. In [5], the authors explore the ability of these three metrics (CK, MOOD, QMOOD) suites to predict fault-prone classes. [5] used 6 versions of Mozilla’ Rhino, compared individual the MOOD and the QMOOD metrics to the CK metrics to see if they are better at predicting initial quality in OO classes. Also, the metrics are examined from all three metrics suites together to determine whether they measure different dimensions of OO class quality or are measuring the same thing. Finally, models using the different metric suites to predict faults are developed. The authors discovered that: 1. the CK and the QMOOD metrics are useful in developing quality classification models to predict defects. 2. The UBLR analysis indicated that the individual metrics CK-WMC, CK-RFC, QMOOD-CIS (Class Interface Size), and QMOOD-NOM (Number of Methods) are consistent predictors of class quality (error-proneness). 3. The MBLR analysis indicated that the CK metrics suite produced the best three models for predicting OO class quality (error-proneness), followed closely by one QMOOD model. 4. The CK metrics have been shown to be better and more reliable predictors of fault-proneness than the MOOD or QMOOD metrics. Like [5], we also want to validate if the MOOD metrics are capable of predicting fault-proneness. In our study, we are going to use the CK metrics to measure fault-proneness of software, since the CK metrics have been proved to be useful to indicate fault-proneness in [1][5]. Then we compare the results computed by the MOOD metrics and the CK metrics to find correlations to examine if the MOOD metrics also effectively predict fault-proneness. Paper [5] only validated the capability of four of the MOOD metrics (AHF, MHF, AIF and MIF). In our study, we will examine all MOOD metrics (include CF and PF) in order to completely evaluate the MOOD metrics on fault-proneness predicting.

Many new metrics specific to the OO paradigm have been developed. However, not enough attention is being paid to sound theoretical foundations and validation of metrics. In [6], the author evaluated capability of the MOOD metrics on a theoretical level. By comparing with the definitions, [6] showed the MOOD team's incomplete and inaccurate understanding of the OO paradigm. Paper [6] also showed that any empirical validation is premature due to the majority of the MOOD metrics being fundamentally flawed. The MOOD metrics either fail to meet the MOOD team's own criteria or are founded on an imprecise, and in certain cases inaccurate, view of the OO paradigm. This paper [6] concludes that the MOOD metrics are imperfect and have the potential to mature into a decent, base-line set of metrics for OO systems. [6] and our study both focus on validation of capability of the MOOD metrics. Paper [6] test if the MOOD metrics have sound theoretical foundation. We are trying to examine if the MOOD metrics can be used to predict error-proneness and change-proneness of software systems. Paper [6] shows that the MOOD metrics have a number of fundamental flaws. However, all the work in [6] is on theory-level without the support of experiments. In our study, we compute MOOD metrics on open source project JFreeChart and analysis the results to test our hypotheses.

III. Metrics

In this section, we introduce the metrics used in our project and their formal definitions. Also we present the challenges we meet during the implementations of these metrics.

A. *Metrics Definition*

In our project, we use all MOOD metrics and some of the CK metrics including DIT, RFC, NOC and CBO.

1) Method Hiding Factor (MHF)

MHF is defined formally as:

$$\frac{\sum_{i=1}^{TC}\sum_{m=1}^{M_d(C_i)}(1-V(M_{mi}))}{\sum_{i=1}^{TC}M_d(C_i)}$$

where TC is the total number of classes and Md(Ci) is the number of methods declared in class Ci, and

$$V(M_{mi}) = \frac{\sum_{j=1}^{TC} is\_visible(M_{mi}, C_j)}{TC-1}$$

where Mmi is a method in Ci and

$$is\_visible(M_{mi}, C_j) = \begin{cases} 1 & iff \quad j \neq i \wedge C_j \text{ may call } M_{mi} \\ 0 & otherwise \end{cases}$$

Mmi is visible to another class Cj if Cj may call Mmi. MHF measures how methods are encapsulated in the classes of a system.

2) Attribute Hiding Factor (AHF)

AHF is defined in an analogous fashion as MHF, but measuring attributes rather than methods. It measures how attributes are encapsulated in the classes of a system.

3) Method Inheritance Factor (MIF)

The Method Inheritance Factor (MIF) metric is defined as follows:

$$\frac{\sum_{i=1}^{TC}M_i(C_i)}{\sum_{i=1}^{TC}M_a(C_i)}$$

where

$$M_a(C_i) = M_d(C_i) + M_i(C_i)$$

and

$M_d(C_i)$
= the number of methods declared in a class,
$M_a(C_i)$
= the number of methods that can be invoked in association with C_i,
$M_i(C_i)$
= the number of methods inherited (and not overridden) in C_i.

MIF measures the degree of inheritance of classes. In the implementation of MIF, we don't consider constructors of classes.

4) Attribute Inheritance Factor (AIF)

AIF has the same definition as MIF except AIF considers attributes instead of methods.

5) Coupling Factor (CF)

CF is defined formally as:

$$\frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{TC} is\_client(C_i, C_j)\right]}{TC^2 - TC}$$

where

$$is\_client(C_c, C_s) = \begin{cases} 1 & iff \quad C_c \Rightarrow C_s \wedge C_c \neq C_s \\ 0 & otherwise \end{cases}$$

CC => CS represents the relationship between a client class CC, and a supplier class CS. The client is either calling a method or accessing an attribute of the supplier class. CF measures the coupling between the classes of a system. In the computation of CF, we don't consider constructors of classes.

6) Polymorphism Factor (PF)

PF was proposed to measure the degree of method overriding in the class inheritance trees. It is formally defined as follows:

$$\frac{\sum_{i=1}^{TC}M_o(C_i)}{\sum_{i=1}^{TC}[M_n(C_i)\times DC(C_i)]}$$

where

$$M_d(C_i) = M_n(C_i) + M_o(C_i)$$

and

$M_n(C_i)$
= the number of new methods,
$M_o(C_i)$
= the number of overriding methods,
$DC(C_i)$
= the descendants count

(the number of classes descending from C_i).

PF is an indirect measure of the relative amount of dynamic binding in a system. We don't consider constructors in the computation of PF.

7) Depth of Inheritance (DIT)

DIT of a class refers to the maximum length from the class to the root of the inheritance tree. It is a measure of how many ancestor classes can potentially affect a class.

8) Response for a Class (RFC)

RFC is defined as follows: RFC = |RS|, where RS is the response set for the class. The response set for the class can be expressed as

$$\text{RS} = \{M\} \cup_{\text{all } i} \{R_i\}$$

where $\{R_i\} =$ set of methods called by method i and

$\{M\} =$ set of all methods in the class.

9) Number of Children of a Class (NOC)

NOC is the number of immediate subclasses subordinated to a class in the class hierarchy. It is a measure of how many subclasses are going to inherit the methods of the parent class.

10) Coupling between Object Classes (CBO)

CBO of a class is a count of the number of other classes to which the class is coupled. It measures the degree of how a class is coupled to other classes.

B. Metrics Implementation details

In our project, we implement the MOOD metrics. During the implementation, the most difficult part is to understand what functionality JDeodorant can provide. In order to understand the exact functionality of methods, we first look for the methods that may be useful in the metric implementation and guess the functionality of these methods according to their names. Then we create a project with few simple classes and use this project to test if our guessing is right or not. After the testing, we get the methods we need for the metric implementation.

To verify if the values of the metrics are correctly computed by our implementation, we first, for each metric, compute its value manually on the project we create. Then we use the same project as input to run our implementation and examine if the generated result is the same as the values we calculate manually.

IV. Empirical Study

In this section, we provide a high-level description of our study. We describe the internal and external metrics used for our study and the hypotheses we want to validate, then we present the experiments for collecting the data. Finally, we analysis the results to verify our hypotheses and discuss the threats that may affect our results.

A. Examined variables

We use the MOOD metrics, including Method Hiding Factor (MHF), Attribute Hiding Factor (AHF), Method Inheritance Factor (MIF), Attribute Inheritance Factor (AIF), Coupling Factor (CF) and Polymorphism Factor (PF) as the internal metrics of our study. And we use some of the CK metrics, including Depth of Inheritance (DIT), Response for a Class (RFC), Number of Children of a Class (NOC) and Coupling between Object Classes (CBO) as the external metrics, since these metrics have been proved to be useful as quality indicators.

B. Examined hypotheses

In our project, we would like to validate the following two hypotheses:

1. Classes with poor values of MOOD metrics are more error-proneness.

2. Classes in latter versions of projects are less error-proneness which means their values of MOOD metrics are better.

For the first hypothesis, we compute values of DIT, RFC, NOC, CBO and the MOOD metrics for each version of JFreechart. Since DIT, RFC, NOC, CBO have been proved to be useful as indicators of error-proneness, we find correlations between these metrics and the MOOD metrics to validate if the MOOD metric can also be used to predict error-proneness. If the first hypothesis is true, we can continue verify our second hypothesis. The values of the MOOD metrics on multiple versions of JFreeChart can be observed in order to test if classes in latter version of JFreeChart have better values of MOOD metrics.

C. Experiment design

We consider 21 versions (from version 1.0.0 to 1.0.19) of JFreeChart in our project. JFreeChart is a large open source project which has 57 versions and many development documents. It is big enough for data collection. And the long history of JFreeChart allows us to better conduct our empirical study and produce more convincing results. The 21 versions we use are the latest versions of JFreeChart.

For each version of JFreeChart, we compute its values of the MOOD metrics, DIT, RFC, NOC and CBO. Depending on the data, we perform two statistical tests, the Pearson coefficient and the linear regression, to help analysis the results. The Pearson coefficient can be used to decide if the independent variables and the dependent variables are linearly associated and the linear regression measures the distance between the independent variables and the dependent variables.

D. Data collection

To compute the MOOD metrics on JFreeChart, we use our implementation. For DIT, RFC, NOC and CBO, we use ckjm, which a tool is used for automatically calculating the CK metrics for Java projects, to collect values. Then for each pair of metrics (one from the internal metrics and the other from the external metrics), we compute the Pearson coefficient and the linear regression by using Excel.

E. Statistical analysis

In this section, we present the results of the Pearson coefficient and the linear regression on the values of the internal and external metrics. Then we analyze the results and present our findings.

1) Pearson Coefficient

The Pearson coefficient is a measure of linear association between two variables. For a pair (xi, yi), i = 1..N, the coefficient R is computed as follows:

$$R = \frac{1}{N-1}\sum_{i=1}^{N}\frac{x_i-\mu_x}{\sigma_x}\;\frac{y_i-\mu_y}{\sigma_y}$$

where μx and σx denote the mean and the standard deviation for variable x, and same for variable y. For each metric in the MOOD metrics and in DIT, RFC, NOC and CBO, Table 1 shows their coefficient values.

Correlation	DIT	RFC	NOC	CBO
MHF	-0.838	0.821	0.634	0.555
AHF	-0.832	0.834	0.631	0.581
MIF	-0.834	0.760	0.629	0.481
AIF	-0.818	0.711	0.609	0.428
PF	0.559	-0.230	-0.404	-0.016
CF	0.839	-0.812	-0.633	-0.540

Table. 1 Values of Correlation Coefficient R

R falls between [-1, 1]. We define the levels of strengths of correlation based on R in Table 2.

	Negative	Positive
None	-0.09 to 0.0	0.0 to 0.09
Small	-0.29 to -0.1	0.1 to 0.29
Medium	-0.69 to -0.3	0.3 to 0.69
Strong	-1.0 to -0.7	0.7 to 1.0

Table. 2 Strength of Correlation Coefficient R

We can get the following conclusions.

I. MHF has a strong negative correlation with DIT, a strong positive correlation with RFC and medium positive correlations with NOC and CBO.

II. AHF has a strong negative correlation with DIT, a strong positive correlation with RFC and medium positive correlations with NOC and CBO.

III. MIF has a strong negative correlation with DIT, a strong positive correlation with RFC and medium positive correlations with NOC and CBO.

IV. AIF has a strong negative correlation with DIT, a strong positive correlation with RFC and medium positive correlations with NOC and CBO.

V. PF has a medium positive correlation with DIT, a small negative correlation with RFC, a medium negative correlation with NOC and no correlation with CBO.

VI. CF has a strong positive correlation with DIT, a strong negative correlation with RFC, and medium negative correlation with NOC and CBO.

2) Linear Regression

Linear regression calculates an equation ŷ = ax + b that minimizes the distance between the dependent variable x and the independent variable y. a and b is computed as follows.

$$a = \frac{\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y)}{\sum_{i=1}^{N}(x_i-\mu_x)^2}$$

$$b = \mu_y - a\mu_x$$

Depending on the equation, a regression line can be drew. R-squared is a statistical measure of how close the values of dependent and independent variables are to the regression line. The closer the values are to the regression line, the more associated the variables are. Following is the steps to compute R-squared.

I. The total sum of squares:

$$\text{TSS} = \sum_{i=1}^{n}(y_i-\bar{y})^2$$

where ȳ is the mean of all values of y.

II. The sum of squares of residuals :

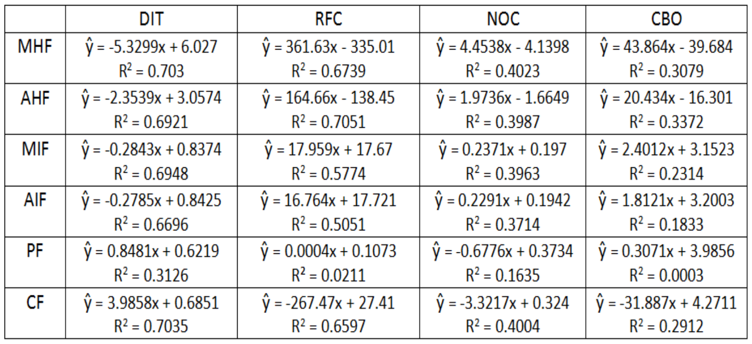
$$\text{RSS} = \sum_{i=1}^{n}(y_i-\hat{y}_i)$$

where ŷ = ax + b.

III. R-squared:

R2 = 1 - RSS/TSS

Table 3 shows the equation and the values of R^2 for each pair of dependent and independent variables.



	DIT	RFC	NOC	CBO
MHF	ŷ = -5.3299x + 6.027 R^2 = 0.703	ŷ = 361.63x - 335.01 R^2 = 0.6739	ŷ = 4.4538x - 4.1398 R^2 = 0.4023	ŷ = 43.864x - 39.684 R^2 = 0.3079
AHF	ŷ = -2.3539x + 3.0574 R^2 = 0.6921	ŷ = 164.66x - 138.45 R^2 = 0.7051	ŷ = 1.9736x - 1.6649 R^2 = 0.3987	ŷ = 20.434x - 16.301 R^2 = 0.3372
MIF	ŷ = -0.2843x + 0.8374 R^2 = 0.6948	ŷ = 17.959x + 17.67 R^2 = 0.5774	ŷ = 0.2371x + 0.197 R^2 = 0.3963	ŷ = 2.4012x + 3.1523 R^2 = 0.2314
AIF	ŷ = -0.2785x + 0.8425 R^2 = 0.6696	ŷ = 16.764x + 17.721 R^2 = 0.5051	ŷ = 0.2291x + 0.1942 R^2 = 0.3714	ŷ = 1.8121x + 3.2003 R^2 = 0.1833
PF	ŷ = 0.8481x + 0.6219 R^2 = 0.3126	ŷ = 0.0004x + 0.1073 R^2 = 0.0211	ŷ = -0.6776x + 0.3734 R^2 = 0.1635	ŷ = 0.3071x + 3.9856 R^2 = 0.0003
CF	ŷ = 3.9858x + 0.6851 R^2 = 0.7035	ŷ = -267.47x + 27.41 R^2 = 0.6597	ŷ = -3.3217x + 0.324 R^2 = 0.4004	ŷ = -31.887x + 4.2711 R^2 = 0.2912

Table. 3 Linear Regression

R^2 falls between [0, 1]. We define the degree of how the equation fits the data based on the values of R^2 in Table 4. Higher value of R^2 indicates that the equation better fits the data and the independent and dependent variables are more associated.

R-squared	
None	0.0 to 0.09
Low	0.1 to 0.29
Medium	0.3 to 0.69
High	0.7 to 1.0

Table. 4 Degree of Fitness

We discuss each metric in the MOOD metrics on linear regression.

I. MHF is highly associated with DIT and is moderately associated with RFC, NOC and CBO.

II. AHF is highly associated with RFC and is moderately associated with DIT, NOC and CBO.

III. MIF is moderately associated with DIT, RFC and NOC and is lowly associated with CBO.

IV. AIF is moderately associated with DIT, RFC and NOC and is lowly associated with CBO.

V. PF is lowly associated with DIT and is not associated with RFC, NOC and CBO.

VI. CF is highly associated with DIT and is moderately associated with RFC, NOC and CBO.

3) Discussion

The results of the Pearson coefficient and linear regression show that only MHF, AIF and CF have strong positive (or negative) correlations and are highly associated with DIT. This means only MHF, AIF and CF can used as the indicators of error-proneness like DIT. Since the larger DIT is, the more error-proneness, the larger values of MHF and AIF mean the less error-proneness and the larger value of CF means the more error-proneness. Therefore, our first hypothesis, that assumes classes that have poor values of MOOD metrics are more error-proneness, is not accurate. Also we observe the values of MHF, AIF and CF for 21 versions of JFreeChart shown in Fig.1, Fig.2 and Fig.3. Some of the previous versions of the project have higher MHF and AIF than the latter versions which means these latter versions are more error-proneness. Also some latter versions have higher CF than the previous versions which means the latter versions are more error-proneness. These are against our second hypothesis that assumes latter versions are less error-proneness than previous versions of the project. Thus, both of our hypotheses are rejected.

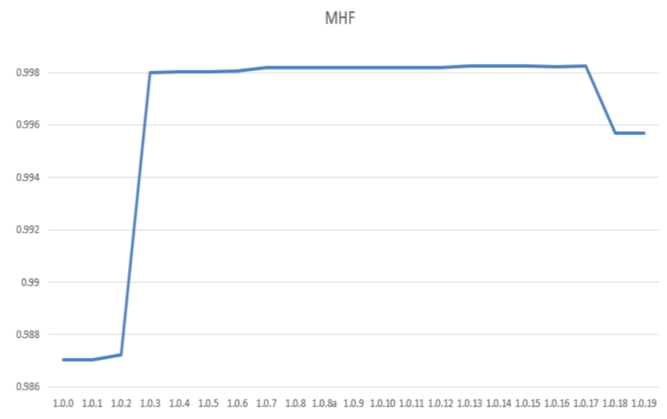


Fig.1 Values of MHF

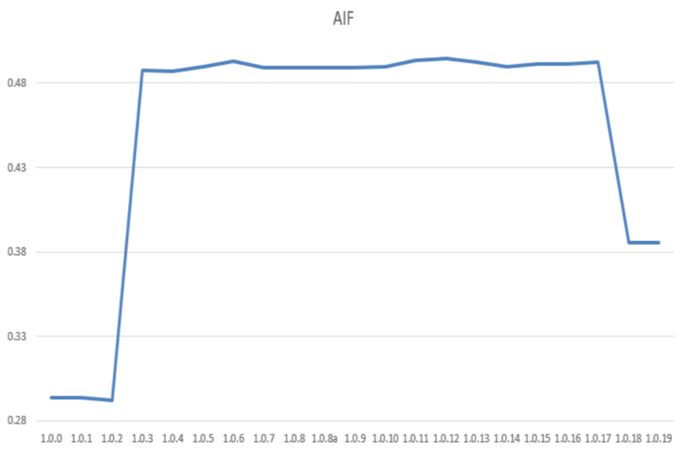


Fig.2 Values of AIF

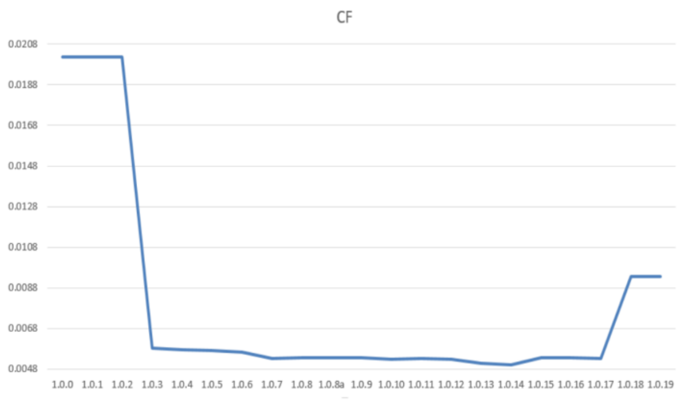


Fig.3 Values of CF

F. *Threats to validity*

We use ckjm for computing our external metrics. Although ckjm is wildly used, there are not validations to verify the correctness of results. In our project, we assume that the results computed by ckjm is correct.

In our implementation of MIF, CF and PF, we don’t consider constructors of classes. It is possible that people may want to consider constructors for these three metrics. In that case, the values of these metrics computed by our implementation may not be accurate.

V. Conclusions

In our study, we examine the associations between the internal metrics and the external metrics. The results of the Pearson coefficient show that internal metrics MHF and AIF have strong negative correlations with external metric DIT and the internal metric CF has strong positive correlation with DIT. The results of the linear regression show that MHF, AIF and CF have are highly associated with DIT. Therefore, MHF, AIF and CF can be used to predict error-proneness as DIT. Since the larger DIT is, the more error-proneness, the larger values of MHF and AIF mean the less error-proneness and the larger value of CF means the more error-proneness. Depending on the values of these metrics on 21 versions of JFreeChart, we can also see that some latter versions of JFreeChart are more errorprone than the previous versions. Therefore, we conclude that only MHF, AIF and CF are useful to predict error-proneness and the error-proneness of the latter versions of JFreeChart are not necessarily better than the previous versions.

REFERENCES

[1] V. R. Basili, L. C. Briand, and W. L. Melo, A Validation of Object-Oriented Design Metrics as Quality Indicators, IEEE Transactions on Software Engineering, vol. 22, no. 10, pp. 751-761, 1996.

[2] Rachel Harrison, Steve J. Counsell, and Reuben V. Nithi, An Evaluation of the MOOD Set of Object-Oriented Software Metrics, IEEE Transactions on Software Engineering, vol. 24, no. 6, pp. 491-496, June 1998.

[3] S.R. Chidamber and C.F. Kemerer, A Metrics Suite for Object Oriented Design, IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476493, June 1994.

[4] Fernando Brito e Abreu WalcClio Melo, Evaluating the Impact of Object-Oriented Design on Software Quality, International Software Metrics Symposium, 1996:90-99.

[5] Hector M. Olague, Letha H. Etzkorn, 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, June 2007.

[6] Tobias Mayer & Tracy Hall, Measuring OO systems: a critical analysis of the MOOD metrics, Technology of Object Oriented Languages and Systems, 1999.

[7] Jagdish Bansiya and Carl G. Davis, A Hierarchical Model for Object-Oriented Design Quality Assessment, IEEE Transactions on Software Engineering, vol. 28, no. 1, pp. 4-17, January 2002.

[8] Steve Counsell, Stephen Swift, and Jason Crampton, The interpretation and utility of three cohesion metrics for object-oriented design, ACM Transactions on Software Engineering and Methodology, vol. 15, no. 2, pp. 123-149, April 2006.