Study of Fault Predictions in Open Source Software

Vishnusaranya¹, Harish Ramkumar², Vivek³

1 P.G Scholar, Dept of Computer Applications, Tamilnadu College of Engineering, Coimbatore, India.
2 P.G Scholar, Dept of Computer Applications, Tamilnadu College of Engineering, Coimbatore, India.
3 Lecturer, Dept. of Computer Applications, Tamilnadu College of Engineering, Coimbatore, India.

Abstract— In this review paper, it is intended to summarize the Columbus framework for computing object-oriented metrics and these metrics are used to measure the quality of the open source software projects and also used to analyze and compare the results with the different versions of open source software projects. In particular, the CK (Chidamber and Kemerer) metrics suite and Columbus framework with compiler wrapping technology were studied and metrics are used to compare the results of several versions of Mozilla. It present the initial set of findings to validate the usefulness of CK metrics for predicting fault proneness in the software.

Keywords— Columbus framework, Compiler Wrapping, Logistic Regression, Machine Learning.

I. INTRODUCTION

The improvement of software process in software development can play a vital role in the delivery of application and information technology [1]. Studies show that traditional product metrics are not sufficient for characterizing, assessing and predicting the quality of object-oriented systems [2]. Object-oriented design metric suites provide the set of metrics to improve the software process and it validates the quality of the software [1]. The study in statistical analysis of Mozilla’s Rhino project indicate that CK metrics have been shown to be better and more reliable predictors of fault-proneness than the MOOD or QMOOD metrics in traditional as well as highly iterative or agile software development process [3]. The CK suite consists of six object-oriented metrics [1].

<table>
<thead>
<tr>
<th>TABLE 1</th>
<th>CK METRICS SUITE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Metrics</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>WMC (Weighted Methods per Class)</td>
<td>The number of methods in a class</td>
</tr>
<tr>
<td>DIT (Depth of Inheritance Tree)</td>
<td>The DIT measures the number of ancestors of a class.</td>
</tr>
<tr>
<td>RFC (Response For a Class)</td>
<td>This is the number of methods that can potentially be executed in response to a message being received by an object of that class.</td>
</tr>
<tr>
<td>NOC (Number Of Children)</td>
<td>This is the number of direct descendants for each class</td>
</tr>
</tbody>
</table>

II. OPEN SOURCE SOFTWARE-MOZILLA

Open-source software is very often developed in a public, collaborative manner [5]. In this study we used web and email suite of Mozilla’s version from 1.0 to 1.6. These are analyzed in Columbus REE (Reverse Engineering Environment) . The company may have risks when open source software were incorporated because of quality and reliability of the code may be uncertain. By the usefulness of various code measures fault-proneness and quality improved [4].

![Diagram](http://www.ijctttjournal.org)

Fig. 1 Overview of section

III. REVERSE ENGINEERING

A. Columbus Framework Systems

The Columbus Framework – a reverse engineering tool developed in cooperation between the university of Szeged, the Nokia Research Center, and FrontendART. The tool
performs various tasks of reverse engineering such as project handling, data extractions, data representation, data storage, filtering and visualization [6]. These tasks are accomplished using appropriate modules (plug-ins) of the system [6]. The intention to the development of Columbus framework is to creation of toolset which supports fact extraction and various reverse engineering tasks [4].

Fact extraction is used to extracting the fact, that describes the different properties of the system. It involves various processing steps [4].

- Acquiring project / Configuration Information
- Analysis of the source code
- Linking
- Filtering
- Processing
- Acquiring Project / Configuration Information:

In this step it captures the source code information of system from the project files and makefiles to proceed the extraction process.

- Analysis of the source code:
  For the every single input file it preprocessed and then analyze using CANPP and CAN tool which produces finally schema instance.

- Linking:
  Using CANLink tool the related schema instances are merged and produces single schema instance to represent the system.

- Filtering:
  If large projects produce large schema instances then filtering methods are applied to extract the data. For filtering Columbus REE uses CANFilter tool.

The filtering options are,

- Filtering by input source files(only classes)
- Filtering according to scope (classes and namespaces)
- Filtering using class dependencies ( Inheritance)
- Filtering by hand (select / deselect the class in a displayed class diagram) [6].

C. Computation Of Metrics

The computation value derived from the Mozilla [4].

<table>
<thead>
<tr>
<th>Version</th>
<th>No. of Classes</th>
<th>LOC</th>
<th>No. of Methods</th>
<th>No. of Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>4770</td>
<td>1127391</td>
<td>69474</td>
<td>47,428</td>
</tr>
<tr>
<td>1.1</td>
<td>4823</td>
<td>1145470</td>
<td>70247</td>
<td>48,070</td>
</tr>
<tr>
<td>1.2</td>
<td>4686</td>
<td>1154685</td>
<td>70803</td>
<td>46,695</td>
</tr>
<tr>
<td>1.3</td>
<td>4730</td>
<td>1151525</td>
<td>70805</td>
<td>47,012</td>
</tr>
<tr>
<td>1.4</td>
<td>4967</td>
<td>1171503</td>
<td>72096</td>
<td>48,389</td>
</tr>
<tr>
<td>1.5</td>
<td>5007</td>
<td>1169537</td>
<td>72458</td>
<td>47,436</td>
</tr>
<tr>
<td>1.6</td>
<td>4991</td>
<td>1165768</td>
<td>72314</td>
<td>47,608</td>
</tr>
</tbody>
</table>

Using the Hopkins scale the CK metrics such as RFC and WMC correlated in a strong manner to find class defects [7].

D. Fault Prediction Techniques

The 20 % of defects leads to the 80% of unavoidable rework and it may be higher for large software system[8]. By constructing the prediction models the faulty classes are identified early in the development [9]. These prediction models are based on historical data, using object-oriented design metrics faulty classes are identified in future applications [9].

As Khaled El Emanet.al study they hypothesise the relationship between fault-proneness and object-oriented metrics based on the cognitive complexity [9].

B. Compiler Wrapping

An issue in the fact extraction step at Acquiring Project / Configuration Information was not to make changes in the subject system. This issue can be resolved using new technique which is known as “Compiler Wrapping”. By wrapping the compiler such that original compiler was temporarily hide by changing path variable using CANGcc Wrapper toolset. This toolset which uses script if it not needed then it can be temporarily switched off.

Cognitive complexity is the mental burden who deals with software components (Coupling) [9]. When these metrics have higher values then the probability of failure can increased in that class.
TABLE 3  
METRICS VALUES FOR MOZILLA

<table>
<thead>
<tr>
<th>Version</th>
<th>WMC</th>
<th>DIT</th>
<th>RFC</th>
<th>NOC</th>
<th>LCOM</th>
<th>CBO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>235</td>
<td>13</td>
<td>843</td>
<td>0</td>
<td>27233</td>
<td>50</td>
</tr>
<tr>
<td>1.1</td>
<td>238</td>
<td>13</td>
<td>852</td>
<td>0</td>
<td>27939</td>
<td>50</td>
</tr>
<tr>
<td>1.2</td>
<td>250</td>
<td>13</td>
<td>885</td>
<td>0</td>
<td>30805</td>
<td>51</td>
</tr>
<tr>
<td>1.3</td>
<td>256</td>
<td>14</td>
<td>902</td>
<td>0</td>
<td>32262</td>
<td>51</td>
</tr>
<tr>
<td>1.4</td>
<td>295</td>
<td>15</td>
<td>1005</td>
<td>0</td>
<td>42529</td>
<td>57</td>
</tr>
<tr>
<td>1.5</td>
<td>337</td>
<td>17</td>
<td>1071</td>
<td>0</td>
<td>55200</td>
<td>56</td>
</tr>
<tr>
<td>1.6</td>
<td>337</td>
<td>17</td>
<td>1074</td>
<td>0</td>
<td>55198</td>
<td>57</td>
</tr>
</tbody>
</table>

E. Logistic regression Model

The logistic regression model which is used to perform defect predictions [12]. This model outputs a value between 0 and 1 when the change occurs and moreover threshold value (i.e. 0.5) is also used. The model predicted probability of a defect is greater than 0.5 the change is classified as defect inducing else non-defect inducing

To predict the evaluation performance the measures such as precision, accuracy, recall are computed from confusion matrix [12].

<table>
<thead>
<tr>
<th>Classified as</th>
<th>True Class</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Defect</td>
<td></td>
<td>Non-Defect</td>
</tr>
<tr>
<td>Defect</td>
<td>TP</td>
<td></td>
<td>FP</td>
</tr>
<tr>
<td>Non-Defect</td>
<td>FN</td>
<td></td>
<td>TN</td>
</tr>
</tbody>
</table>

\[
\text{Precision} = \frac{TP}{TP+FP} \\
\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \\
\text{Recall} = \frac{TP}{TP+FN}
\]

F. Machine learning models

Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data, or both [13]. The decision tree is a classic and natural model of learning. It is closely related to the fundamental computer science notion of “divide and conquer.

The decision tree is so-called because we can write our set of questions and guesses in a tree format, the questions are written in the internal tree nodes (rectangles) and the guesses are written in the leaves (ovals). Each non-terminal node has two children: the left child specifies what to do if the answer it to the question is “no” and the right child specifies what to do if it is “yes” [14].

By using machine learning models such as neural network, decision tree to the prediction of bugs with the help of CK metric suites. The faultiness can be accessed from the class using metric values of open source software.

IV CONCLUSION

In this paper a method or technique to preprocess the file, which was used in the Columbus framework. The detailed explanation about various metrics used to analyze the various versions of open source software. The information of the paper deal with predictions about defect by analyzed versions of open source software are studied and explained.

We are currently performing initial analysis with various versions of open source software such as Columbus, rhino and Mozilla to predict faultiness in class.

ACKNOWLEDGMENT

The authors would like to thank his team members and guide, for the assistance of helping in to analyze open source software along with different tools.
REFERENCES


[10] CAS MA 116 • Statistics I • Summer 2, 2012 • Lecture 5.3 “Logistic regression Analysis”


